O'REILLY®

TURING 图灵程序设计丛书

学习响应式设计

LEARNING RESPONSIVE WEB DESIGN

[美] Clarissa Peterson 著

文巍 译



中国工信出版集团 人民邮电出版社 POSTS & TELECOM PRESS

数字版权声明



图 6-8：因为颜色数不够，这张 GIF 图像看上去是失真的



图 6-9：屏幕上一幅图像中的单个像素放大后的样子



图 6-10：以实际尺寸显示的图像



图 6-11：大于实际尺寸显示的图像



图 6-22：背景图

Colors

Colors are used to color code sections, define lower level content and direct attention to important content.

Primary Colors



Dark Blue
Background colors used in the masthead
#0678BE



Light Blue
Used for smaller body copy, used in the masthead
#53B0EB



Lime Green
Used for download buttons and to indicate customization
#96BC44

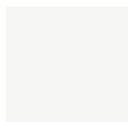
Secondary Colors



Navy Blue
Background color used in the top navigation
#064771



Light Navy Blue
Background color used in the lower navigation
#0D7DC1



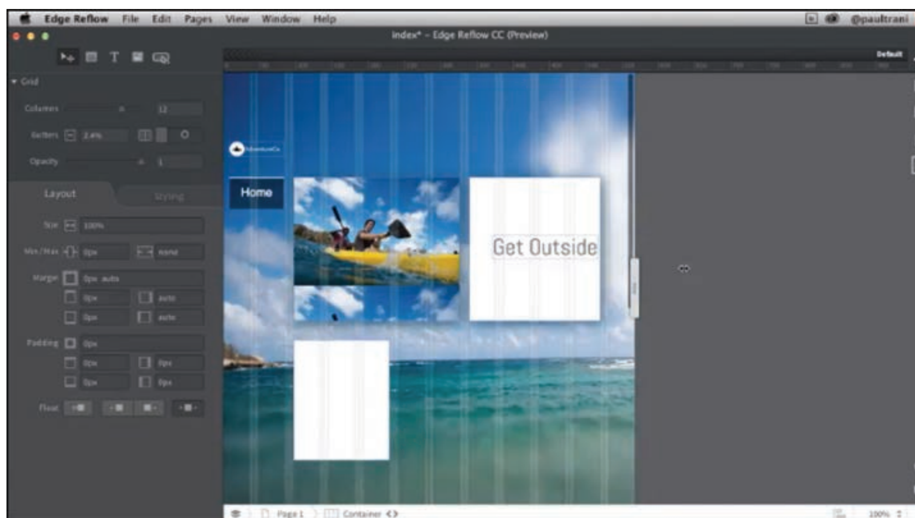
Pale Gray
Background color used in the footer, advertising and to highlight important information.
#F6F6F2







图 7-10：Drupal.org 风格指南中的一页列出了颜色（主要颜色、次要颜色及信息颜色）；每个条目给出颜色名、十六进制值及其用途（比如，深蓝色，#0678BE，用作刊头的背景色）



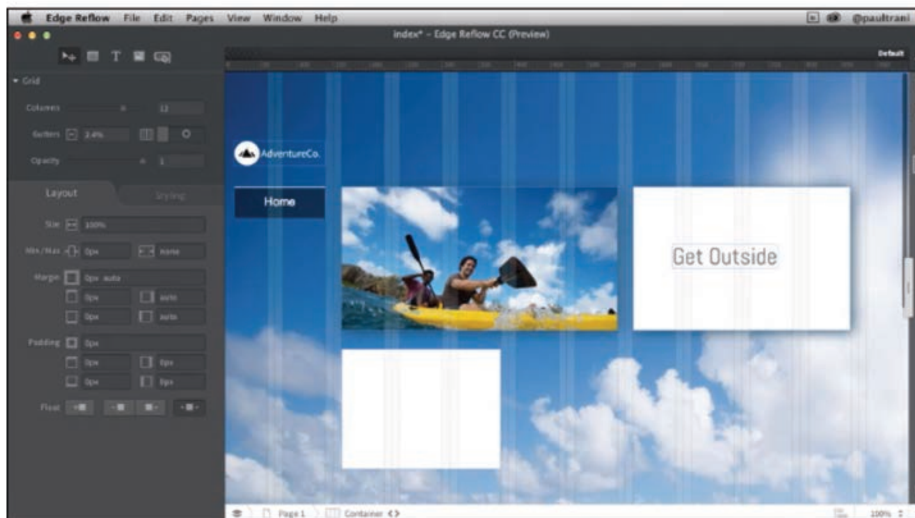


图 7-12：Adobe Edge Reflow



图灵程序设计丛书

学习响应式设计

LEARNING RESPONSIVE WEB DESIGN

［美］Clarissa Peterson 著

文巍 译

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

O'Reilly Media, Inc.授权人民邮电出版社出版

人民邮电出版社

北　　京

图书在版编目（CIP）数据

学习响应式设计 /（美）彼得森（Peterson,C.）著；文巍译. -- 北京：人民邮电出版社，2015.6
（图灵程序设计丛书）
ISBN 978-7-115-38973-2

Ⅰ. ①学… Ⅱ. ①彼… ②文… Ⅲ. ①计算机网络—程序设计 Ⅳ. ①TP393.09

中国版本图书馆CIP数据核字(2015)第072083号

内 容 提 要

本书分为四部分，全面介绍响应式Web设计策略与技术，引领大家掌握从项目立项到网站上线的响应式设计工作流程。第一部分介绍响应式设计及其与其他网页设计方法的区别。第二部分概述构建响应式站点的基础知识、必备的HTML和CSS知识、图像处理，以及响应式设计的核心：媒体查询。第三部分带大家全面了解响应式设计的工作流程，从用户体验的角度探讨响应式设计，确保网站在各种设备上的适应性和功能完备性。第四部分深入探讨需特别考虑的设计元素，包括文字排版、响应式导航和页头，以及响应式设计的一大问题：性能。

本书适合所有从事网站相关工作的人员阅读参考，包括但不限于前端开发人员、Web设计师、产品经理。

◆ 著　　　[美] Clarissa Peterson
译　　　文　巍
责任编辑　李松峰　毛倩倩
执行编辑　李舒扬
责任印制　杨林杰

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路11号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址：http://www.ptpress.com.cn
北京　　　　　　印刷

◆ 开本：800×1000　1/16
印张：20.5　　　　彩插：2
字数：487千字　　　　2015年6月第1版
印数：1 - 4 000册　　　　2015年6月北京第1次印刷

著作权合同登记号　图字：01-2014-8585号

定价：69.00元

读者服务热线：(010)51095186转600　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京崇工商广字第 0021 号

版权声明

O’Reilly Media, Inc.介绍

O’Reilly Media 通过图书、杂志、在线服务、调查研究和会议等方式传播创新知识。自 1978 年开始，O’Reilly 一直都是前沿发展的见证者和推动者。超级极客们正在开创着未来，而我们关注真正重要的技术趋势——通过放大那些“细微的信号”来刺激社会对新科技的应用。作为技术社区中活跃的参与者，O’Reilly 的发展充满了对创新的倡导、创造和发扬光大。

O’Reilly 为软件开发人员带来革命性的“动物书”；创建第一个商业网站（GNN）；组织了影响深远的开放源代码峰会，以至于开源软件运动以此命名；创立了 Make 杂志，从而成为 DIY 革命的主要先锋；公司一如既往地通过多种形式缔结信息与人的纽带。O’Reilly 的会议和峰会集聚了众多超级极客和高瞻远瞩的商业领袖，共同描绘出开创新产业的革命性思想。作为技术人士获取信息的选择，O’Reilly 现在还将先锋专家的知识传递给普通的计算机用户。无论是通过书籍出版，在线服务或者面授课程，每一项 O’Reilly 的产品都反映了公司不可动摇的理念——信息是激发创新的力量。

业界评论

“O’Reilly Radar 博客有口皆碑。”

——*Wired*

“O’Reilly 凭借一系列（真希望当初我也想到了）非凡想法建立了数百万美元的业务。”

——*Business 2.0*

“O’Reilly Conference 是聚集关键思想领袖的绝对典范。”

——*CRN*

“一本 O’Reilly 的书就代表一个有用、有前途、需要学习的主题。”

——*Irish Times*

“Tim 是位特立独行的商人，他不光放眼于最长远、最广阔的视野并且切实地按照 Yogi Berra 的建议去做了：‘如果你在路上遇到岔路口，走小路（岔路）。’回顾过去 Tim 似乎每一次都选择了小路，而且有几次都是一闪即逝的机会，尽管大路也不错。”

——*Linux Journal*

目录

前言 ······ XV

第一部分　响应式设计基础

第 1 章　什么是响应式设计 ······ 3

1.1　一点儿基础 ······ 3
1.2　简史 ······ 6
1.2.1　固定宽度设计 ······ 6
1.2.2　移动 Web 浏览 ······ 7
1.2.3　移动网站 ······ 8
1.2.4　更多的设备 ······ 9
1.2.5　媒体查询 ······ 10
1.2.6　灵活性 ······ 11
1.2.7　响应式网页设计 ······ 12
1.3　为什么是响应式设计 ······ 12
1.3.1　每个设备都得到正确的设计 ······ 12
1.3.2　更少的工作 ······ 14
1.3.3　搜索优化 ······ 14
1.4　总结 ······ 15

第 2 章　响应式内容 ······ 16

2.1　内容策略 ······ 16
2.2　内容整理 ······ 18
2.2.1　仅使用你需要的 ······ 18

2.2.2 如何精简 ······19
2.2.3 内容审计 ······20
2.3 内容编写 ······20
2.3.1 用户如何阅读 ······20
2.3.2 简明的语言 ······22
2.4 内容平等 ······23
2.5 内容管理 ······24
2.6 自适应内容 ······24
2.7 总结 ······27

第二部分　创建响应式网站

第 3 章　响应式网站之 HTML ······31
3.1 使用 HTML ······32
3.1.1 HTML 版本 ······32
3.1.2 Web 标准 ······33
3.1.3 使用 HTML5 ······33
3.2 页面基本结构 ······34
3.2.1 文档类型声明 ······34
3.2.2 文档结构 ······34
3.2.3 页面标题 ······35
3.2.4 字符集 ······36
3.3 视口 ······37
3.3.1 宽度 ······40
3.3.2 初始缩放比 ······40
3.3.3 用户可缩放 ······40
3.3.4 最大缩放比 ······41
3.4 结构性元素 ······41
3.4.1 屏幕阅读器 ······42
3.4.2 <header> ······43
3.4.3 <nav> ······43
3.4.4 <footer> ······43
3.4.5 <article> ······44
3.4.6 <aside> ······44
3.4.7 IE 支持 ······44
3.5 创建一个页面 ······45
3.5.1 结构性元素 ······45

3.5.2 加入内容 ······ 46
3.5.3 无样式页面 ······ 47
3.6 清晰和语义化的 HTML ······ 48
3.6.1 分离内容与表现 ······ 48
3.6.2 注释 ······ 49
3.7 总结 ······ 50

第 4 章 响应式网站之 CSS ······ 51

4.1 CSS 的工作方式 ······ 52
4.2 CSS 版本 ······ 53
4.3 置 CSS 于何处 ······ 57
4.3.1 嵌入式样式 ······ 57
4.3.2 样式表 ······ 58
4.3.3 内联样式 ······ 59
4.4 层叠 ······ 59
4.4.1 层叠的作用方式 ······ 59
4.4.2 重要的规则 ······ 60
4.4.3 内联样式 ······ 60
4.4.4 ID 选择器 ······ 60
4.4.5 类、属性和伪类选择器 ······ 61
4.4.6 元素与伪元素选择器 ······ 61
4.4.7 继承规则 ······ 61
4.4.8 默认值 ······ 62
4.4.9 发生冲突怎么办 ······ 62
4.5 使用层叠 ······ 62
4.5.1 默认值和重置 ······ 63
4.5.2 继承规则 ······ 64
4.5.3 元素规则 ······ 64
4.5.4 其他规则 ······ 65
4.5.5 保持简单 ······ 65
4.6 注释 ······ 66
4.7 组织你的样式表 ······ 66
4.8 盒模型 ······ 67
4.8.1 度量单位 ······ 68
4.8.2 em ······ 68
4.8.3 高度与宽度 ······ 69
4.8.4 外边距与内边距 ······ 70
4.8.5 边框 ······ 71

4.8.6 盒子大小 72
4.9 显示 74
4.10 定位 75
4.10.1 静态定位 75
4.10.2 相对定位 75
4.10.3 绝对定位 76
4.10.4 固定定位 77
4.10.5 定位元素的度量单位 78
4.11 浮动和清除 79
4.12 基本样式 81
4.13 总结 83
第 5 章 媒体查询 85
5.1 何为媒体查询 85
5.2 媒体查询结构 87
5.3 在样式表链接中使用媒体查询 89
5.4 使用媒体查询的其他方式 90
5.5 我们可以查询什么 91
5.5.1 视口宽度和高度 91
5.5.2 屏幕宽度与高度 92
5.5.3 方向 92
5.5.4 宽高比 92
5.5.5 分辨率 93
5.5.6 其他媒体特性 93
5.6 浏览器支持 94
5.6.1 IE 条件注释 94
5.6.2 测试媒体查询结果 96
5.7 断点 96
5.8 设计范围 97
5.9 响应式设计 98
5.9.1 渐进增强 98
5.9.2 用网格进行设计 99
5.9.3 使用网格列 100
5.9.4 优先为小屏幕设计 101
5.10 使用媒体查询 102
5.11 两列布局 102
5.11.1 使用浮动 103
5.11.2 使用网格 105

5.11.3 加入媒体查询 …… 107
5.12 设置最大宽度 …… 108
5.13 如何选择断点 …… 109
5.14 总结 …… 110

第 6 章 图像 …… 111

6.1 显示图像的方式 …… 112
6.1.1 CSS 替代方案 …… 112
6.1.2 内容图像 …… 113
6.1.3 背景图像 …… 113
6.1.4 图像拼合 …… 113
6.1.5 图标字体 …… 114
6.2 替代文本 …… 115
6.3 图像文件格式 …… 118
6.3.1 JPEG …… 119
6.3.2 GIF …… 119
6.3.3 PNG …… 120
6.3.4 SVG …… 120
6.4 优化图像 …… 121
6.4.1 像素与分辨率 …… 121
6.4.2 高密度屏幕 …… 122
6.4.3 压缩图像 …… 123
6.4.4 实际尺寸 …… 124
6.5 内容图像 …… 126
6.5.1 <img> 元素 …… 126
6.5.2 加入图像 …… 127
6.5.3 灵活的图像尺寸 …… 129
6.5.4 媒体查询 …… 130
6.5.5 最大宽度 …… 132
6.5.6 用图像叙事 …… 134
6.6 背景图像 …… 134
6.6.1 加入背景图像 …… 135
6.6.2 对齐 …… 136
6.7 响应式图像 …… 137
6.7.1 建议的客户端解决方案 …… 138
6.7.2 其他解决方案 …… 139
6.7.3 断点 …… 143
6.8 总结 …… 145

第三部分　玩转响应式设计

第 7 章　响应式设计工作流程 149
7.1 策略与规划 149
7.1.1 用户研究 150
7.1.2 内容 150
7.2 内容先于布局 152
7.2.1 内容组件 152
7.2.2 以文本模式进行设计 153
7.2.3 线性设计 155
7.2.4 内容层次 155
7.3 考虑布局 155
7.3.1 草图绘制 155
7.3.2 从小屏幕开始 156
7.3.3 移动优先 157
7.4 原型 158
7.4.1 线框图与原型 158
7.4.2 线框图 158
7.4.3 响应式原型 159
7.4.4 原型中有什么 160
7.4.5 从基础开始 160
7.4.6 创建页面布局 160
7.4.7 框架 162
7.4.8 原型创建工具 163
7.5 视觉设计 164
7.5.1 样式板 164
7.5.2 测试与调整 165
7.5.3 风格指南 166
7.6 响应式设计工具 168
7.6.1 Adobe Photoshop 168
7.6.2 Adobe InDesign 168
7.6.3 Adobe Edge Reflow 169
7.6.4 Adobe Dreamweaver 170
7.6.5 浏览器 171
7.7 推销响应式设计 172
7.7.1 为什么要用响应式设计 172
7.7.2 教育你的客户 173

7.7.3 强调响应性 173
7.7.4 响应式设计并不总是最佳选择 174
7.7.5 费用 174
7.8 与客户合作 175
7.8.1 交付物 175
7.8.2 陈述报告 176
7.9 总结 177

第 8 章 岂止手机 178

8.1 用户体验 178
8.1.1 用户至上 179
8.1.2 手机用户的谬见 180
8.1.3 为应用环境设计 180
8.1.4 只用手机的用户 181
8.1.5 使用多个设备 182
8.2 设备无关的设计 182
8.3 专注于移动优先 183
8.4 尽你所能 183
8.5 设备类型 186
8.5.1 手机 186
8.5.2 平板电脑 187
8.5.3 其他设备 187
8.5.4 台式电脑与笔记本电脑 188
8.6 触摸 188
8.6.1 电容式触摸 189
8.6.2 多点触摸 189
8.6.3 手势 189
8.6.4 JavaScript 事件 190
8.6.5 触摸目标大小 191
8.6.6 导航位置 193
8.7 屏幕尺寸 194
8.8 可访问性（通用化设计） 196
8.8.1 视觉 197
8.8.2 音频 199
8.8.3 输入方法 200
8.8.4 认知障碍 201
8.9 决定支持哪些设备 201
8.10 为何要用真实设备进行测试 202

8.10.1 设备实验室 203
8.10.2 购买设备 203
8.11 测试 203
8.11.1 验证器 204
8.11.2 在浏览器窗口调整 204
8.11.3 浏览器工具 204
8.11.4 浏览器与操作系统 205
8.11.5 仿真器与模拟器 206
8.11.6 辅助技术 207
8.12 总结 207

第四部分 设计响应式网站

第 9 章 文字排版 211
9.1 始于 HTML 212
9.2 字型 212
9.3 使用字体 214
9.3.1 设计良好的字体 214
9.3.2 自托管字体 214
9.3.3 字体服务 215
9.3.4 链接字体文件 216
9.3.5 创建字体栈 217
9.4 调整文字大小 218
9.4.1 忘掉像素 218
9.4.2 屏幕距离 218
9.4.3 绝对与相对 219
9.4.4 设置默认字体大小 220
9.4.5 为什么是 100% 220
9.4.6 度量单位 221
9.4.7 字体大小间的关系 225
9.4.8 行高 225
9.5 行长 227
9.5.1 测试行长 228
9.5.2 调整外边距及字体大小 229
9.5.3 断字 231
9.5.4 溢出换行 232
9.6 留白 233

9.7 内边距和外边距……234
9.8 为屏幕尺寸改变字型……234
9.9 总结……235

第 10 章 导航及页眉布局……237

10.1 响应式导航……237
10.1.1 从小屏开始……238
10.1.2 样式化列表……238
10.1.3 水平导航……240
10.2 网站品牌……243
10.3 导航链接……245
10.3.1 灵活性……245
10.3.2 用户想要做什么……245
10.3.3 基于目标的导航……247
10.3.4 保持一致性……248
10.3.5 保持简单……250
10.4 导航模式……250
10.4.1 顶部导航……252
10.4.2 页脚导航……254
10.4.3 切换式推出型导航……256
10.4.4 切换式覆盖型导航……260
10.4.5 部分优先型导航……260
10.4.6 选择菜单型导航……261
10.4.7 抽屉式导航……263
10.4.8 底部导航……264
10.4.9 跳过子导航……266
10.4.10 被抛弃的导航……267
10.4.11 用于宽屏幕的固定菜单……269
10.5 页眉……270
10.5.1 极简页眉……270
10.5.2 复杂页眉……271
10.5.3 导航图标……273
10.5.4 其他图标……275
10.6 总结……277

第 11 章 性能……279

11.1 性能的重要性……280
11.2 性能作为设计元素……280
11.2.1 网络连接……281

11.2.2 平衡 281
11.2.3 臃肿的网络 281
11.3 网页加载与渲染方式 282
11.3.1 延迟 283
11.3.2 DNS 请求 283
11.3.3 重定向 284
11.3.4 HTTP 请求 284
11.3.5 发送 HTML 文件 284
11.3.6 解压 285
11.3.7 DOM 285
11.3.8 渲染 <head> 元素 286
11.3.9 渲染 <body> 元素 287
11.3.10 onload 事件 287
11.4 性能测量 287
11.5 清理代码 289
11.5.1 使用简单直接的代码 289
11.5.2 缩小 290
11.6 将 HTTP 请求减至最少 291
11.6.1 串联 291
11.6.2 第三方代码 292
11.6.3 图像拼合 293
11.7 服务器设置 293
11.7.1 避免重定向 293
11.7.2 文件压缩 294
11.7.3 浏览器缓存 295
11.8 JavaScript 296
11.8.1 JavaScript 做什么 296
11.8.2 JavaScript 的工作方式 296
11.8.3 阻塞式 JavaScript 297
11.8.4 JavaScript 库 300
11.9 CSS 301
11.10 托管 302
11.10.1 内容分发网络 302
11.10.2 内容管理系统 303
11.11 有条件地加载内容 303
11.12 重绘与回流 305
11.13 RESS 305
11.14 总结 307

前言

2007 年，iPhone 的发布对于网站设计来说是一个转折点，网站设计师们突然失去了对画布（我们在上面设计网站）的控制。此前，网站只是在显示器屏幕上供人浏览，尽管屏幕尺寸有所不同，但差别并不是那么大。而现在，我们怎样才能使网站在 iPhone 这么小的屏幕上供人浏览呢？

在开始的一段时间里，我们制作了专门针对 iPhone 屏幕尺寸进行了优化的移动网站，其独立于我们的“常规”网站。同时维护两个站点还不是那么糟，但大量不同屏幕尺寸的手机很快涌现出来，然后是平板电脑以及更小号的平板电脑，我们终于意识到不可能为每种可能的屏幕尺寸都制作一个网站。

我们需要一种适用于所有屏幕尺寸的解决方案，需要一种设计网站的方法，使网站能够自动适配各种显示屏幕。

一时间各种创意层出不穷，不久之后响应式网站设计脱颖而出。它是一种灵活的不依赖于固定屏幕尺寸的网站设计方法，能够检测屏幕的大小并调整设计，从而针对具体设备提供最佳的视觉体验。2010 年，Ethan Marcotte 在 *A List Apart* 上首次发表了关于响应式网站设计的文章（http://alistapart.com/article/responsive-web-design）。

如其他新技术一样，响应式网站设计一开始并不被人接受，许多人不断辩称——甚至于现在某些人还在这么做——我们需要为手机单独创建网站。但随着今天市场上设备的激增，很明显，我们不能依赖于一种手机模型并将其作为设计标杆。我们的设计要能够适应所有设备，而这些设备的屏幕尺寸各不相同。

而与此同时，响应式设计也在不断发展。它不再仅仅是适应屏幕尺寸，也能适应不同的设备性能，如触摸屏、视网膜显示屏，以及慢速连接。

在 2014 年，58% 的美国成年人拥有一部功能丰富并允许用户完全访问网络的智能手机，

这些手机搭载了 iOS、Android、Windows Phone 等操作系统，[1] 且 35% 的美国成年人拥有一部平板电脑。[2] 我们有令人惊叹的硬件设备，而响应式设计能帮助我们充分利用网络。

然而，尽管大多数美国成年人都有智能手机，但仍有 32% 受访者的手机不是智能手机。这之中的很多人用手机访问网络时只能使用功能受限的浏览器，这种浏览器无法如他们希望的那样显示所有网站。响应式网站设计也是这种情况下的一种解决方案。

响应式网站设计一开始只是设计简单的、注重内容的站点（不依赖 CSS 或 JavaScript），使其能显示在几乎所有的联网设备上。在此基础之上，这种设计通过渐进式增强——响应式网站设计正是基于此——针对具体的显示屏尺寸以及设备功能进行优化。因而，功能手机（也即功能有限的旧式手机）只能获得那些它们能使用的内容，而相对来说较新的智能设备所访问的网站则具有丰富的设计、完美适配其屏幕的界面，设备功能被充分利用。

响应式网站设计使我们有能力为所有用户呈现最好的网站，而不管他们具体使用什么设备。网络对每个人应该都是可用的，而响应式设计将指导我们做到这一点。

创建响应式网站并不仅仅是学习几段新代码那么简单。它还包括重新审视网站构想方式，关注用户体验，并确保内容和功能不是在设计完成之后才加上去的。

我们还必须改变网站制作方式，过渡到一个涉及设计师、开发者和其他团队成员的更具协作性的过程中去。

我们需要学习一些新的代码，但响应式设计并不是一种新的编程语言，创建一个响应式网站只需要 HTML、CSS，有时再加点 JavaScript。如果你已经知道如何制作网站，那么本书中讲到的大部分内容对你来说都会很熟悉。你需要记住，在创建一个响应式网站时，90% 的工作与创建非响应式网站时是一样的。但除了增加少许新技术，你需要掌握正确的基础知识，使用结构恰当、合乎规范的标记（HTML 和 CSS）。如果没有坚实的基础，你难以确信自己的网站能正确工作并在各种设备上恰当地显示。

如果你从事与建设网站相关的工作，不管是一名 Web 设计师、开发人员、内容策划、用户体验设计师、网站负责人、IT 主管，还是从事其他任何涉及创建和维护网站的工作，你都将从本书中知晓：响应式设计的工作原理、怎样调整自己的工作流程以适应响应式设计，以及如何创建对任何设备都能提供最佳设计与用户体验的响应式网站。

注 1：完整报告请查看 Susannah Fox 和 Lee Rainie 发表的“The Web at 25 in the U.S.”，皮尤互联网研究项目，2014 年 2 月 27 日（http://www.pewinternet.org/2014/02/27/the-web-at-25-in-the-u-s/）。

注 2：更多信息请查看 Lee Rainie and Aaron Smith 发表的“Tablet and E-reader Ownership update 2013”，皮尤互联网研究项目，2013 年 10 月 18 日（http://www.pewinternet.org/2013/10/18/tablet-and-e-reader-ownership-update/）。

组织结构

本书分为四部分。

第一部分解释了什么是响应式设计，以及它与其他网站设计方法有何不同。我们也将介绍如何创建弹性内容，并确保它们能很好地工作于响应式网站。

第二部分概述了构建响应式站点的基础知识。我们将学习一小部分 HTML 和 CSS 知识，它们是让网站正确工作的关键。然后，我们深入介绍响应式设计的核心——媒体查询。最后，我们会看看在响应式网站中如何处理图像。

第三部分带大家看看响应式设计的工作流程，即按部就班创建一个响应式网站的过程，它涵盖了从项目启动会议开始直至网站上线运行的内容。之后我们将进一步从用户体验的角度探讨响应式设计，审视如何确保网站能在不同的输入方式（比如触摸）下工作，以及如何确保网站对所有用户都是功能完好的，包括那些使用辅助技术的用户。

最后，第四部分深入研究那些对于响应式网站需特别考虑的设计元素。我们首先从文字排版开始讲起，这是确保内容在不同尺寸的屏幕上都可读的关键。然后，我们看看如何编写响应式导航和页头。最后，我们会讨论响应式设计的一个大问题——性能，因为我们尝试让用户即使在低速连接上也能在合理的时间内加载完网站。

目标读者

本书适合每一位网站从业人员阅读参考，任何经验水平的人都能够读懂本书。

如果你是一名已经非常熟悉 HTML 和 CSS 的开发人员，那么其中一些内容对你来说一定不陌生——响应式网站设计与非响应式网站设计有很多相同之处（但区别总是存在的）。如果你在这方面没有任何经验，也不必担心，本书将会带领你从最基本的知识开始学习，同时从设计角度综述响应式设计的方方面面。

如果你从未使用过 HTML 和 CSS，那么必须去全面理解构成一个响应式网站所必需的代码，以及它们是如何工作的。不过，本书并不是一本 HTML 或 CSS 入门书，所以其中每个概念的介绍都相当简洁，不涉及太多细节。如果想学习 HTML 和 CSS，你应该参考其他书籍和资源以进行更深入地学习。但如果你在工作中并不需要编写实际的代码，只是想了解响应式设计的工作原理，本书也定将满足你的需求。

如果你介于新手和老手之间，本书将提示你制作一个网站的所有事项，并向你展示响应式设计与非响应式设计的不同之处。你所看到的不仅是代码，还有对设计注意事项的考量以及响应式设计的工作原理。

联系我们

请把对本书的意见和疑问发给出版社。

美国：

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472

中国：

北京市西城区西直门南大街 2 号成铭大厦 C 座 807 室（100035）
奥莱利技术咨询（北京）有限公司

O'Reilly 的每一本书都有专属网页，你可以在那儿找到本书的相关信息，包括勘误表、示例代码以及其他信息。本书的网站地址是：

http://shop.oreilly.com/product/0636920029199.do

你可以在这里下载本书的所有代码示例：

http://www.learningrwd.com/

对于本书的建议和技术性问题，请发送电子邮件到：

bookquestions@oreilly.com

要了解更多 O'Reilly 图书、培训课程、会议和新闻的信息，请访问以下网站：

http://www.oreilly.com

我们在 Facebook 的地址如下：http://facebook.com/oreilly

请关注我们的 Twitter 动态：http://twitter.com/oreillymedia

我们的 YouTube 视频地址如下：http://www.youtube.com/oreillymedia

致谢

首先最想感谢的是 Scott Berkun，他在 2012 年于 An Event Apart Seattle[3] 上所做的演讲促使我辞去了办公室的工作，从而能够去做更有意义的事情。阅读 Scott 的《演讲之禅：一位技术演讲家的自白》（英文版由 O'Reilly 出版）一书后，我开始尝试做关于响应式设计的演讲，这给我带来了很多机会。Scott 还鼓励和建议我在写书这条道路上走下去，并把我介绍给了他在 O'Reilly 出版社的编辑（也成了我这本书的编辑）。没有 Scott，我很可能仍郁郁寡欢地呆在某个办公室隔间中，你也就不会读到这本书了。

注 3：An Event Apart 是为期两天的 Web 设计会议，面向所有拥有激情的从业者。

感谢 O’Reilly 出版社的 Mary Treseler 及其他参与本书出版过程的所有工作人员。

在本书写作期间，我结了婚并移居到了一个新的国家。我的丈夫 A.J. Kandy，对于在这些重大的人生变化中还稀里糊涂地想着要写书的我一直抱有极大的耐心。我何其幸运能与这世上最棒的男人结为伴侣。

当然，也非常感谢 Ethan Marcotte，没有他提出的响应式设计理念，我也不会在今天想要写作一本这方面的书。

感谢 Matt Bradley 很早之前教我 HTML。

有几个人在我的职业生涯中竭尽所能地帮助我朝正确的方向前进，极其感谢他们的帮助：Dan Brown、Jeffrey Zeldman、Vera Rhoads 和 Cory Lebson。

也要感谢一路以来许多给我帮助，给我鼓励或建议的人们：Theresa Amato、Melissa Ballowe、Chuck Borowicz、Erica Ciesielski Chaikin、Glennette Clark、Sibyl Edwards、Veronica Erb、Brad Frost、Robert Hoekman, Jr.、Tim Kadlec、Dave Mankoff、Karen McGrane、Jeff Popovich、Chris Schmitt、Jared Spool、Anna Storkson、Sandy Tetreault、Estelle Weyl。当然，还要感谢我的父母。

感谢所有在聚会和其他团体中认识的人们，我有幸与他们一起讨论响应式设计。我开车周游美国东北部和中西部时玩得很开心，还认识了那么多了不起的人。希望将来能去更多的地方旅行，因为我想要结识太多的人。

第 一 部 分

响应式设计基础

第 1 章

什么是响应式设计

现在，网络世界中的所有人几乎都听说过响应式网页设计（通常称为 RWD，即 Responsive Web Design 的缩写），但大部分人并没有很好地理解它是什么。

在本章中，你将学习响应式网站的基础知识。之后，我们将简述网页设计的简史，以便你理解响应式设计的思想来自哪里，以及它与旧网页设计方式之间的差异。

我们还会讨论为什么对于制作能很好地工作于不同设备与屏幕尺寸的网站而言，响应式设计通常是最佳选择，以及为何从长远来说它意味着更少的工作。我们也会介绍选择响应式设计的一个不太明显的效果：它对你的搜索引擎排名的影响。

1.1 一点儿基础

如果你拿起本书，是因为你听说过响应式设计，但还不是很明白它到底是什么，本节将帮助你了解响应式设计的基本知识。

即使你有一定的响应式设计经验，也可能觉得很难用通俗的语言向别人解释它。本节将给你一个更好的主意，告诉你如何向用户、客户、非技术团队成员或你的妈妈（她很好奇，想知道你每天在工作中都做些什么）解释响应式设计。

总的来说，响应式设计是一种方法，使网站可以在任何类型的设备和任何尺寸的屏幕上（从最小的手机直至最宽的桌面显示器）轻松浏览和使用。

最简单的解释方式是比较响应式网站与非响应式网站，并看看这两种类型的网站在智能手机上的浏览效果。

想象你正使用你的智能手机浏览一个固定宽度的网站，即一个被设计成总是以一个固定的宽度（比如 960 个像素）来显示的站点。你将看到整个网站就如同它在你的桌面显示器上显现的一样，但它最初以一个微小的尺寸进行显示，以适应屏幕。你频繁地放大和缩小它，以便阅读文字和浏览网站，如图 1-1 所示。这需要许多额外的操作。

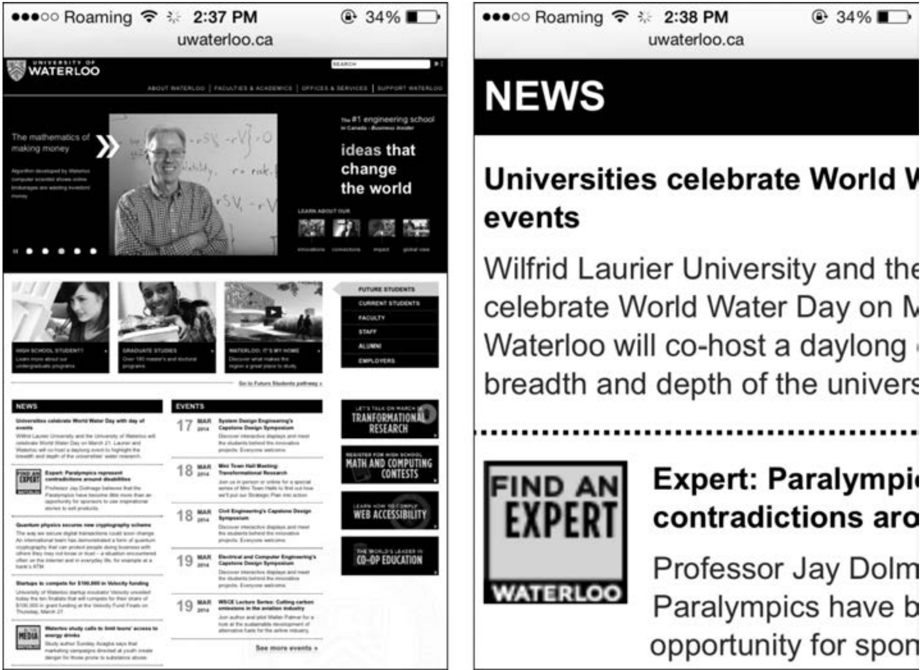


图 1-1：在手机上浏览一个固定宽度的网站时，需要放大文本才能看清

有些网站有一个移动版站点，独立于常规的桌面网站。在你的智能手机上浏览此类网站时，它以全尺寸显示（不必进行缩放），但你时常会发现它与你在桌面显示器上浏览的同一网站有很大不同——通常缺失很多内容，网站拥有者因此可以减少维护多个版本网站所产生的额外工作。

此外移动版网站通常是为一个特定尺寸的设备（如 iPhone）制作的，因此如果你有一个不同的设备，该网站可能就无法很好地适应屏幕了。

单独的移动网站通常是为某一大小的设备所优化的，但是市场上有很多不同的设备，构建一个只能工作于一种设备上的移动网站可能意味着抛弃了所有使用其他不同设备的用户。

举一个例子，宜家（IKEA）有一个单独的移动网站，是为某一尺寸的手机优化适配的。在图 1-2 中，上方是宜家桌面站点的导航栏，左下方是该站点在 iPad 中的样子，右下方是该站点在 iPhone 中的样子。

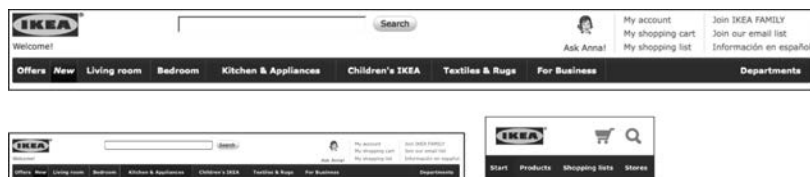


图 1-2：宜家桌面站点（上方）与你在 iPad 上（左下）所看到的是一样的，而 iPhone 则显示一个特别的移动网站（右下）

三个屏幕截图都是按比例缩小的，你可以比较在不同尺寸屏幕上所看到的内容。在 iPhone 上浏览的是移动版网站，只显示几个导航链接，但它们与桌面网站上的链接有相似的尺寸大小。而在 iPad 上，你浏览的是桌面网站而不是移动版网站，所有内容都被缩得非常小以适应小屏幕。你必须做许多缩放操作来浏览该网站。

宜家公司做了大量工作来创建一个良好的移动网站，但如果你的设备是一台平板电脑，你没法使用它，也不会获得满意的体验。如果宜家有一个响应式网站，就能够保证人们使用任何尺寸的设备都能获得一个合适的界面。

采用响应式设计时，网站仅有一个版本，因此你能浏览全部的内容，且网站会自己重新排列以完美地适配任何尺寸的屏幕，并具有全尺寸文本，那样你就无需进行缩放操作了，如图 1-3 所示。

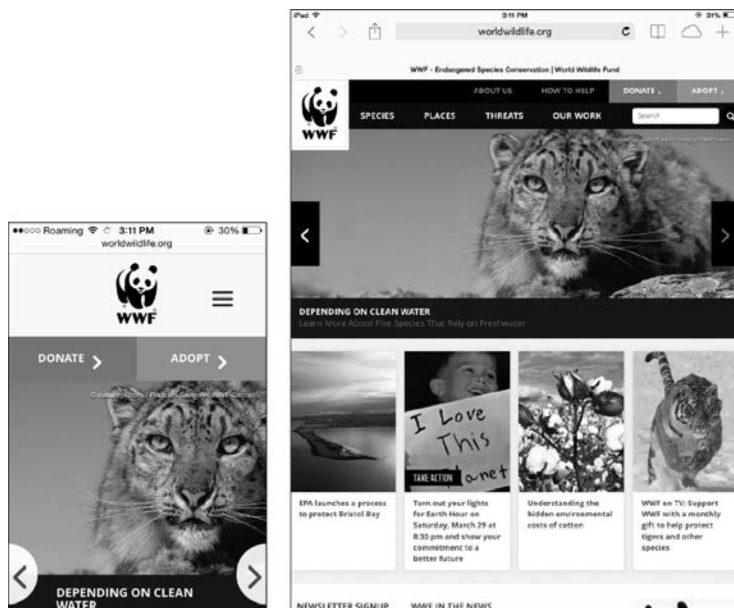


图 1-3：在手机、平板及桌面显示器上浏览一个响应式网站

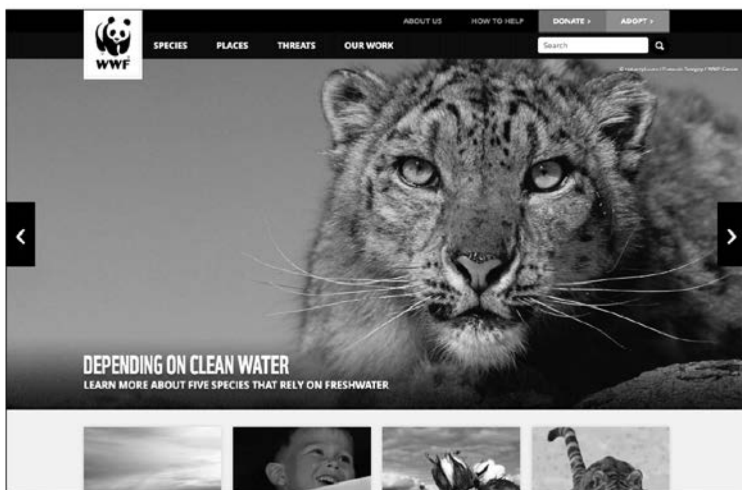


图 1-3 （续）

这里有很多技术细节（稍后我们将讨论），但从用户的角度来说，描述一个响应式网站的关键是网页内容可以自动改变大小和移动位置以适配显示它们的屏幕。

1.2　简史

在学习响应式设计是如何产生的之前，了解一点网页设计的历史是有帮助的。

1.2.1　固定宽度设计

几年以前，网站是被设计成固定宽度的，以便很好地适配最常见尺寸的台式机和笔记本的屏幕。在 2000 年，这意味着设计的屏幕宽度是 800 像素（一个像素是屏幕上一个彩色发光的小点），到 2005 年前后，大多数显示器是 1024 像素宽。

尽管大多数显示器通常就是那几种固定的尺寸，但市场上还存在一些更宽的显示器，也有些老式的窄屏显示器仍在使用。网页设计师希望他们的设计在所有显示器上看起来都完全相同，所以他们通常会设计一个固定宽度的网站以适应最常见的显示器尺寸，如 960 像素宽的网站在 1024 像素宽的屏幕上无疑是适合的。在更大的屏幕上，网站则会简单地以空区域（设计术语为空白）来填充两边的额外空间，如图 1-4 所示。



图 1-4：固定宽度的 MSN 站点在任何尺寸的屏幕上都是相同的宽度，当屏幕比站点宽时两边会留出空白

这种固定宽度的设计如今仍是相当普遍的。

流动设计（fluid design）和*流式布局*（liquid layout）的想法在 21 世纪初吸引了不少注意力。这些技术所采用的基于百分比的宽度使得设计的网页能以流式方式适应屏幕的宽度。因此它可以充分利用大屏幕的可用空间。尽管这在理论上听起来不错，但这种放弃对宽度控制的设计最终意味着，在宽屏幕上网站将变得超级宽而且栏目分得非常开，这令大多数 Web 设计师仍然坚持使用更容易处理的固定宽度的设计。

1.2.2　移动Web浏览

手机首次能访问因特网是在 20 世纪 90 年代中期，它们通常并不具备显示实际网站的能力，而只能显示文本数据，比如天气预报、股票报告和比赛得分。最初的移动浏览器只能显示基本的 HTML 格式，且通常是黑白而不是彩色。直到 2005 年左右，在更先进的称为*智能手机*的设备上（比如图 1-5 中所示的 iPhone），移动浏览器才能够显示使用了 CSS2 和 JavaScript 技术的“真正的”网页。



图 1-5：iPhone 4S

2007 年发布的 iPhone 改变了游戏规则。它有一个漂亮的界面，可以利用当时所有的 Web 技术显示网页，如它们在全尺寸显示器上所显示的一样。

虽然触屏设备已经存在多年，但 iPhone 是第一个采用*多点触控*技术的移动设备，能同时识别屏幕上的多个触点——这对我们现在习以为常的诸如捏合以缩放等浏览行为来说是必不可少的。

所有这一切令 iPhone 不断被评为那一年最具革命性的产品之一，但仍有一个问题存在。网页是设计来在全尺寸显示器上而不是在小手机上浏览的，因此几乎所有网页都是 960 像素或者更宽。另一方面，iPhone 的屏幕只有 320 像素宽。苹果的解决方案是自动收缩网页以适应屏幕的浏览区域（*视口*），然后允许用户通过双击或捏合手势缩放页面的任何区域。

但一次只能浏览页面的一小块无法产生好的用户体验。

网页设计师知道，这对于使用网络的人们来说不是一个最佳方式，他们需要找到一种方法使得网页能更容易地在 iPhone 的小屏幕上浏览。

1.2.3　移动网站

因为设计师习惯于制作固定宽度的网页，所以最简单明了的解决方案是制作单独的移动版网站（具有一个固定的页面宽度，适配 320 像素宽的屏幕，而不是常见的 1024 像素宽的显示器），如图 1-6 所示。



图 1-6：华盛顿邮报网站的桌面版和手机版

如果用户使用的是手机，通常会被自动重定向到移动版网站。否则，可以通过点击一个链接来选择去移动版网站，或者通过一个不同的 URL 访问移动版网站，通常使用 m 子域名（比如 http://m.sprint.com）。这种做法使得这些单独的移动版网站被称为 m-dot 网站。

当然，这对 Web 团队意味着额外的工作，但他们通常会简化这项工作，把移动版网站做成标准网站的精简版：只具有一小部分内容，如图 1-6 所示。

他们的理由是：手机只是在“四处奔走”或某些基本活动中才使用。这在当时对大多数用户来说可能是对的。但随着手机越来越普及，人们开始使用这些设备来执行越来越多的任务，而他们之前仅在台式机或笔记本电脑上执行这些任务。

1.2.4 更多的设备

做一个“iPhone 网站”，他们如是说。起初，在 iPhone 是智能手机市场仅有的霸主玩家时，这种方式很好。但没多久，其他手机公司很快紧跟潮流，推出它们的产品以应对 iPhone。

但这些新的智能手机并非都拥有相同的尺寸。相对于 iPhone 的 320 像素宽度，许多拥有更窄的屏幕（240 像素或更少），另一些则拥有比 iPhone 更宽的屏幕（特别是那些被设计成水平握持而不是垂直握持的设备）。320 像素宽的 iPhone 版网站并不能很好地适应那些屏幕。

因此网页设计师开始尝试寻找一种解决方案：我们怎么制作一个网站，使其能用于所有尺寸的屏幕？这没有简单的答案。

到了 2010 年，苹果发布了 iPad，再一次改变了游戏规则。移动版网站太小，不能充分利用 iPad 更大的屏幕空间，但桌面尺寸固定宽度的网站对于以纵向模式浏览的 iPad 来说又显得太大了。

尽管有些设计师的反应是创建单独的 iPad 网站（现在他们已经有三个独立的网站了），但大多数人意识到，随着越来越多的各色尺寸的设备上市，为每一种可能的屏幕尺寸创建单独的网站不再是一项可持续下去的工作。

1.2.5 媒体查询

网页设计社区重拾流式布局的理念，使用基于百分比的宽度，并设法让其成为一种适用于小型移动设备的解决方案。

使用百分比而不是像素使得网页及页面各部分都能够自动改变宽度来适应任意屏幕尺寸，因此也就很容易地消除了相似尺寸设备之间的差异。

但是一旦你纵观所有的设备，就会有这样一个问题。如果要缩窄三栏的布局以应用于智能手机的屏幕宽度，那么各栏中的文本会变得窄而长，难以阅读。同样，单栏布局在智能手机屏幕上看起来是蛮好的，但要想在桌面显示器上轻松阅读则显得太宽了。

本质上，这个问题是：不创建多个单独的站点，如何使一个网站能够在窄屏上以单栏显示，而在大屏上以多栏显示？如何根据浏览网站的设备的特性来要求浏览器变换网页的布局？

媒体查询登场。

CSS 的 @media 规则允许你根据设备特性而显示不同的 CSS 样式，这实际上在十多年前的 CSS2 中就有了，但当时它只支持查询媒体的类型，比如屏幕（screen）或打印（print）。这通常只能用于为网站设计创建一个打印版（可能包含的变化也仅仅是去除背景色，以免浪费打印机墨水）。

直到 CSS3 对媒体查询制定了*规范*（即关于某事必须怎样做的一个正式详尽的描述），才能够基于媒体（设备）的宽度、高度和色彩性能等特性执行更精确的查询。媒体查询不影响 HTML（页面之下的实际内容和结构），只影响使用 CSS 后应用到页面的样式。浏览器在 2009 年前后开始支持 CSS3 媒体查询。

媒体查询可以做什么呢？

举个基本的例子，假设我们有一个网站，有两块各自独立的内容。我们可以创建一个很适合智能手机的单栏设计，以垂直堆叠的方式显示两块内容。而在更宽的屏幕上，我们可能想要以并排的两栏来显示这两块内容。

使用媒体查询，我们可以询问设备屏幕有多宽，然后告诉浏览器仅在屏幕具有足够宽度时，才以两栏来显示内容。

要用代码来实现这种效果，我们在开始只是用 CSS 将内容显示在一栏中，然后再添加一条 CSS 媒体查询，询问屏幕宽度是否大于等于 40 em（你将在第 4 章中了解 em，40 em 的宽度略窄于一台常见的平板电脑，但你可以在媒体查询中指定任一宽度）。

之后我们会在这条媒体查询中添加 CSS 样式将内容显示在两栏中。浏览器仅在该媒体查询结果为真时（即如果屏幕宽度大于等于 40 em）才使用此 CSS 样式。如果屏幕宽度小于 40 em，它会忽略此 CSS 样式，内容保持在一栏中。

因此，我们可以针对不同的屏幕尺寸使网站变换为不同的布局，而无需创建单独的网站。

通过使用媒体查询，我们可以随心所欲地改变网站的样式，不仅仅是栏数。媒体查询可以移动内容，改变文字的大小，隐藏或显示内容块，调整边距和空白，以及调整其他的 CSS 样式。

[小贴士]

在这一节中，我提到了询问设备屏幕宽度的媒体查询。实际上，在响应式设计中常见的媒体查询是询问设备的视口宽度，而不是屏幕宽度。视口是屏幕上（浏览器窗口中）显示网站的区域。

在台式电脑上，你可以改变浏览器窗口的大小，因此，窗口并不总是屏幕的最大宽度。媒体查询会检测浏览器窗口内的空间，所以如果你改变了浏览器窗口的大小，你的视口也会发生变化。在移动设备上，你无法改变窗口的大小，所以屏幕和视口总是相同的宽度。

尽管正确的术语应该是视口宽度，但你仍会经常听到人们在谈论媒体查询和响应式设计时说到屏幕宽度。从技术上讲他们说得并不正确，他们很可能指的是视口宽度（正如我在本书中所做的，除非我明确指出）。

还有其他的媒体查询，可以检测设备屏幕的实际宽度而不是视口宽度，但在目前它们并不常用。

1.2.6 灵活性

媒体查询可以重排你的页面布局，但如果没有灵活性做基础，响应式设计将无法工作。

首先，在你的站点中几乎每一种水平度量都需要采用灵活的单位，而不是僵化的像素。这意味着各栏的宽度和其他布局元素将以百分比定义，文本则以一个叫作 em 的相对单位来定义。

缩放页面上的图像这一工作有点不同，因为你不一定想让它们按照屏幕的宽度来改变大小，而是希望只要屏幕上有空间，图像就可以显示得足够大以便看清细节。可问题是，依据设备屏幕的大小，并不总是有足够的空间来以全尺寸显示一个图像。你需要确保图像在没有足够空间的情况下不会被截断。在第 6 章中，我们将学习一个 CSS 技巧来确保图像总是能够与我们放置它的空间相适应。

1.2.7　响应式网页设计

这些思想（媒体查询或灵活性）本身都不是新的或开创性的思想。但在 2010 年，网页设计师 Ethan Marcotte 琢磨出来一套组合使用这些概念来制作网站的方法，能够自动对不同的屏幕尺寸做出响应。

Marcotte 创造了"响应式网页设计"这个术语，并于 2010 年在一篇为 *A List Apart* 撰写的文章（ http://alistapart.com/article/responsive-web-design）中首次提及，随后在 2011 年其撰写的 *Responsive Web Design* 一书得到出版（http://www.abookapart.com/products/responsive-web-design[1]）。

1.3　为什么是响应式设计

响应式网页设计的概念自首次提出以来一直备受热议。就像任何新技术思想一样，有些人接受它，另一些人则唾弃它。

1.3.1　每个设备都得到正确的设计

使用响应式设计最有说服力的理由是，你创建的网站不仅在现今市场上的各种设备上都能正确地工作并有良好的显示效果，而且很可能在将来出现的那些新设备上也能如此。

此外，使用响应式设计，你不会冒用户可能在桌面显示器上浏览移动版站点的风险，反之亦然。

如果你有多个独立的站点，那这无疑会成为一个问题，无论你是通过设备检测来发送正确版本的站点给每个设备，还是使用一组单独的 URL（比如一个 m-dot 子域名）供移动版网站使用。

如果网站有一个单独的移动版本，通常会使用设备检测（其发生在网站服务器，且在页面显示之前）来确定应该将哪个版本的网页（移动版或桌面版）发送给任一特定的设备。这样，站点中的每个页面虽然只有一个 URL，但实际上存在两个不同版本的 HTML 页面。然而，以上这一过程并不是完全可靠，有时会发送错误的页面版本。此外，设备的检测过程也会增加页面的加载时间。

注 1：中文版参见人民邮电出版社的《移动优先与响应式 Web 设计》。

为单独的移动版站点（即 m-dot 站点）使用不同的 URL 是比较容易实现的，但是这要求用户访问的是正确的网站版本。对于通过社交媒体或电子邮件在用户之间来回传递的链接，获得一个正确版本的页面通常会给用户增加额外的负担，或者有时他们根本没法选择。

例如，如果一个桌面用户将《纽约时报》网站上的一个链接通过电子邮件发给一个手机用户，手机用户访问该链接时，显示出来的页面会在屏幕的上方显示一条信息，告诉他可以切换到网站的移动版，如图 1-7 所示。这看上去不错，但用户需要执行额外的操作，花费额外的时间来点击并加载一个完全不同的页面。



图 1-7：在手机上访问桌面版《纽约时报》网站的一个链接，显示的是桌面版站点并包含一条可切换至移动版的提示消息

另一方面，如果我用手机访问《纽约时报》的移动版网站，并发送一篇文章的链接给某人，他在台式电脑上打开链接时，看到的将是为移动设备优化的移动版页面（如图 1-8 所示），页面中没有清晰明确的方式告之应该如何打开完整的桌面版站点。用户可以读这篇文章，但他必须通过点击来查看图像的全尺寸版本，而且他无法看到存在于桌面版网站中的大量补充链接和推荐文章。



Treasury Puts a Date on When Cash May Run Out: Oct. 17

By ANNIE LOWREY
September 25, 2013

WASHINGTON — The Treasury will only have $30 billion of cash on hand by mid-October, putting the United States on the precipice of an unprecedented default, the department said on Wednesday.

The warning puts additional pressure on a hamstrung Congress, which is already struggling to prevent the federal government from shutting down over a budget impasse on Oct. 1.

In a letter to Speaker John A. Boehner of Ohio, Treasury Secretary Jacob J. Lew warned that a single day's net expenditures can be as high as $60 billion. On any given day after that mid-October deadline, money going out might overwhelm money coming in plus cash on hand.

That means that until Congress lifts its statutory debt ceiling, the Treasury could miss or be forced to delay paying some of its bills. Such an event would be unprecedented, and financial analysts anticipate a violent market reaction that

图 1-8：点击《纽约时报》手机版上的链接将进入手机版的页面，即使你使用的是台式电脑

对于响应式设计，你的网页只有一个版本，所以你永远不会遇上“错误的版本”这个问题。无论在什么设备上浏览该网站，都能得到正确的显示。

1.3.2　更少的工作

使用响应式设计最明显的优势是，无论是网站、设计、代码还是内容，你都只需要创建一份。

如果你的网站有一个单独的手机版站点，你需要创建和维护两套（或更多）完全独立的 HTML 页面。任何改动都需要对每一个站点进行同步修改，即便你试图让它们保持一致，也几乎肯定会出问题，某些事物最终将变得不再匹配。尽管使用内容管理系统（CMS）或模板系统可使这项工作变得容易些，但还是有更多的代码和内容要维护，更多的事情可能会被弄乱。

对于响应式网站，你只有一套内容，无论屏幕大小总能恰当地显示。未来的设计调整也能够通过更改样式表来达成。

对于那些没有响应式设计经验的人，起初创建一个响应式网站（你学会了一切如何工作）可能比创建一个固定宽度的网站要付出更多的努力，但从长远来看，你在网站以后的维护工作中将更轻松。

1.3.3　搜索优化

一个单独的手机网站具有一组独立的 URL，这会影响你的网站在搜索结果中的排名。

如果你有两个不同版本的页面，它们具有相同或相似的内容，但 URL 不同（例如，http://www.example.com 和 http://m.example.com），则搜索引擎需要知道它们是被当作同一个页面，从而使得该页面可以被正确索引并在搜索结果列表中显示为一个条目。

虽然这可以通过使用 JavaScript 或服务器端代码来实现，却显得有点复杂，并且如果你做得不正确，最终可能导致两个版本的页面都出现在搜索结果中，从而给用户造成困惑，还可能对你的搜索排名造成负面影响。

谷歌自 2012 年以来开始推荐将响应式设计用于为智能手机优化的网站，不仅因为它能创建一个更好的用户体验，还因为它允许谷歌的网站爬虫更有效地检索你的内容，这意味着对网站所做的改变将会更快地在搜索结果中得到更新。

必应推荐网站使用必要的方法来确保仅产生一组 URL，但没有明确推荐使用响应式设计来做到这一点。

1.4 总结

在引入响应式设计之前，网站通常都是固定宽度的，这意味着网站的设计无论在多大的屏幕上都是以相同的宽度显示。在智能手机出现后，这就真的行不通了，因为网站在小屏幕上显得很小，用户必须不断地进行缩放来阅读内容。

移动网站紧接着出现了，许多公司独立于它们的主站点（或称为桌面站点）创建了第二个网站。移动网站通常只包含主网站上的一小部分内容和功能，所以手机用户对其并不感冒。

随着越来越多的设备出现，设计师们很快意识到创建多个网站以适应每一种不同尺寸的屏幕，即使不是不可能的，也是不切实际的。响应式设计的概念被引入，它使得网站能够响应设备的宽度，并以一种适合于屏幕尺寸的方式来显示网站的内容。

响应式网页设计包含两个主要部分：灵活性，也就是说水平度量需要使用相对单位，比如百分比，这样它们才能对不同尺寸的屏幕做出响应；媒体查询，其允许你基于设备的屏幕宽度，用 CSS 来改变网站的设计。

响应式设计允许你只用一套代码就能为任意屏幕尺寸的设备提供一个合适的设计。不必维护多套不同的代码意味着更少的工作。同时，应用响应式设计意味着你的网站是为搜索引擎优化了的。

接下来，我们将在第 2 章中介绍内容的重要性，以及如何确保设计出在响应式网站上有良好显示效果的内容。

第2章

响应式内容

如果你认为 HTML 和 CSS 是一个网站最重要的部分，那我们需要从另一个角度来看待这个问题。用户访问你的网站是为了获取内容（及功能），而不是为了欣赏设计师的才华或开发者的编程技术。

用户不会在乎你的网站是否是响应式的，而且他们中的大多数人可能连“响应式网页设计”是什么意思都不知道。他们不会考虑自己是否使用了适当的设备来访问网站，或者他们的屏幕尺寸是否合适。他们甚至通常都不关心网站是什么样子的，只要能用他们自己的设备轻松获得所需的信息或功能就行。

这就是为什么在设计网站时我们要首先考虑内容。

与此相反，如果你先设计，然后再填充内容，那你的内容将永远限于次要状态，你也不太可能为用户提供他们需要的内容。

因此，对于响应式网站，你首先需要考虑的是内容，以确保内容在小屏幕上有良好的显示效果。如果你使用固定宽度网站中的现有内容，将很难将其硬塞进较小的屏幕布局中。如果你从零开始创建新内容，则需要确保它是为所有屏幕尺寸优化的，而不只是一种屏幕尺寸。

2.1　内容策略

“内容策略”（content strategy）一词在过去的几年里获得了极大的关注，特别是在 2009 年 Kristina Halvorson 的著作 *Content Strategy for the Web*（New Riders）出版之后。

在那之前，罕有公司谈及自己网站的内容策略。只有当需要创建内容去填充网站上的待填充区域时，我们才会讨论网站内容。

确实，通常我们在创建网站时完全没有考虑内容策略。如果我们的公司或项目需要一个网站，我们会首先设计一个很漂亮并且与品牌相匹配的网站，接下来列出所有现有的将要放到网站上的东西（内容），然后试着将它们都塞到新设计的网站中去。

如果幸运的话，我们能有一个*信息架构师*，他构建了一个良好的架构来组织我们的所有内容。否则，就是所有的内容都堆砌在那里。无论我们有什么内容，都让它们在网站上的某个地方呈现出来。毕竟，从技术的角度来看，只要花费非常少的成本就能把网站做得更大（拥有更多的内容和页面），那么何不把所有内容都放到网站上，让人们在需要的时候可以随时获取，直到永远呢？

但后来我们突然意识到网站的内容*很重要*。

内容策略指的是规划和管理内容的一种方法。这些内容不仅包括文本，也包括其他形式的信息，如图像、视频和音频。内容策略不仅限于网站上有什么，还关系到网站如何运作。

首先要从大局着眼。网站应该实现什么样的目标？用户访问网站时，你希望他们做什么？什么是成功的用户交互？如何用网站来支持你的商业目标或项目目标？如何满足用户需求？

制定一套内容策略要花费时间和精力，但如果你是在进行视觉设计之前而不是之后考虑内容，那么你创建的网站会更好。

要获得有关内容策略的更多信息，请查阅以下资源：

- *Content Strategy for the Web*，Second Edition，Kristina Halvorson 与 Melissa Rach 合著，由 New Riders 出版（http://contentstrategy.com/）；
- “Content Strategy for Mobile”，Karen McGrane 发表于 *A Book Apart*（http://www.abookapart.com/products/content-strategy-for-mobile）；
- “Content Strategy Within the Design Process”，Brad Shorr 发表于 *Smashing Magazine*（http://uxdesign.smashingmagazine.com/2011/12/02/content-strategy-within-design-process/）。

[小贴士]

如果你不知道“内容策略”“内容营销”“内容管理”之间的区别，也不太了解所有其他与内容相关的术语？可以看看 Melissa Rach 发表在 *UX Magazine* 上的文章“Content Strategy and Its Cousins”(http://uxmag.com/articles/content-strategy-and-its-cousins)。

移动内容策略

近来，由于每个人都在关注如何创建适用于移动设备的网站，“移动内容策略”这个词也开始流行起来。内容策略无疑是重要的，所以移动内容策略肯定是超级重要的，对吧？

并非如此。

记住，移动和非移动设备之间不再有严格的界线。正如你到目前为止在本书中所学的那样，我们的目标是创建能够在所有设备上工作的网站，当然也包括移动设备。

所以你不需要一个额外的移动内容策略。实际上，移动内容策略通常与常规的网站内容策略无异，只是你不能再忽略移动设备了。

你要制定的依然是可靠的内容策略，但请记住，用户将会使用各种设备在不同的场景下访问你的网站，所以你在制定内容策略时一定要考虑到这一点。

要使创建的内容在移动设备上和桌面电脑上呈现出同样好的效果，你会遇到很多从未想过的新挑战。务必现在就开始考虑。

2.2　内容整理

设计一个响应式网站时，你可能一开始就有很多潜在的内容：可能是你旧网站上的内容，线下的内容或者只是关于网站应该提供什么内容的一些想法。

你需要做的第一件事就是思考如何处理这些内容。

2.2.1　仅使用你需要的

首先，你真正需要的内容要远远少于你自以为需要的。

千万不要觉得你应该把所有的内容都放到网站上，以防有人需要。当然，如今在线存储的价格相当便宜，我们认为它几乎是免费的，但是把所有内容都放置在网站上还是有明确成本的。

首先是用户的成本。网站上的内容越多，用户在得到所要的内容之前需要翻找的内容也就更多。

当然，有可能他碰巧在你的网站上又发现了一些感兴趣的内容，但更多时候不是这样。难道会有人需要去看贵公司过去十年发布的所有新闻稿吗？

网站上多余的页面将使导航和搜索结果变得混乱。页面上多余的内容将迫使用户做更多的滚动操作。不必要的东西越多，用户找到其所需就越难，放弃查找的可能性也就越大。

其次是网站所有者的成本。他需要投入更多的资源来跟踪、组织和重新组织、不断更新现有内容，以及持续地检查链接是否失效。你网站上的内容越多，某些内容过时或不正确的的可能性就越大，如果你给出了错误的信息，将要承担潜在的责任。你不能将内容添加到网站上就置之不理，而是需要一直维护。

精简内容的最佳时机是在设计新网站之前，或是重新设计现有网站之前。内容越少，设计过程就越容易，构建网站时所需做的工作也越少。

2.2.2 如何精简

当决定要保留什么内容时，你需要从你的商业目标和用户需求两方面考虑。

例如，“在网站上发布我们所有的新闻稿”不是一个商业目标。相反，目标应该像这样：“确保媒体联系人可以得到撰写有关我们业务的文章所需的信息。”

或许你在网站首页放置了 10 篇最新新闻稿的链接，因为你为自己的成就感到自豪。但那会过多占用宝贵的首页版面。你的绝大多数用户不是新闻界的从业人员，剩下的人也没兴趣阅读你公司的新闻稿。将新闻稿放到一个专用的媒体页既可以确保媒体人士仍能找到它们（他们知道如何查找该页面），同时你也不会强迫其他人去看他们永远不会点击的链接。如果你真的想分享最近的成就，那就写成另一种风格呈献给普通读者：在介绍中使用简短直白的语言，而不是商业辞令。

同时还要考虑每个单独内容块的长度。如果“关于我们”页有 1000 字，那你说的就太多了。它没必要是一篇冗长乏味的公司成长故事（甚至包括了过去 30 年每一个董事会成员的名字）。大多数用户进入该页面想要知道的，非常简单，就是你的网站或公司是做什么的。如果这些信息不能用一两句话说清楚，用户恐怕不会做进一步的阅读。如果你必须列出所有董事会成员，请把它们放在一个单独的页面中。

别忘了留意多余的内容。即使你的 CMS 能很方便地把内容布置于多个地方，但除非真的必要，否则不要这样做。它不仅占用网站的空间，还会干扰用户。这还意味着同样的内容将会在搜索引擎中出现多次，这会让用户感到困惑，如果搜索引擎视这些页面为重复的内容的话，也降低你的搜索排名。

在本书后面的部分，我们将讨论小屏幕优先的设计方法，让你先设计一个用于小屏幕设备的布局，然后再转而为更宽屏幕的设备设计布局。这样的设计方式真的非常有助于你精简内容。当你想在小屏幕布局中填充内容时，它迫使你考虑什么内容是真正重要的，并给你一个清楚的答案，而这在你首先为桌面尺寸的网站设计内容时是很难得到的。

2.2.3　内容审计

当重新设计现有网站时，最好先做一下内容审计，也就是把你目前拥有的所有内容做一下盘点、列出一个清单。如果是一个大型网站，结果可能会让你大吃一惊。

通过查阅清单你可以决定哪些内容需要保留，哪些内容可以舍去，哪些新内容需要创建，以及谁来负责内容的创作与编辑。

一旦你在清单中列出了所有的事项，就能很容易地找到并决定如何处理它们。

清单不需要事无巨细，但至少应该进行分类，同时也要包含主要的内容。如果是一个新网站，你应该给出这样一个清单，它不仅有网站上所计划要包含的内容，也有内容开发的时间进度表。

若想了解更多关于内容审查的知识，可阅读 Donna Spencer 发表在 *UXmastery* 上的文章“How to Conduct a Content Audit”（http://uxmastery.com/how-to-conduct-a-content-audit/）。

2.3　内容编写

一旦你弄清楚了需要哪些内容以及把它们放在哪里，还需要考虑如何具体编写内容。

2.3.1　用户如何阅读

你可曾想过用户是如何在一个网站上阅读的？我们喜欢设想他们会访问网站上的每一个页面并将它们从头读到尾，可实际上并非如此。

“建好了，他们就会来”可不是 Web 内容创作者的座右铭，至少不应该是。不过你很可能被这样的网站愚弄过，这种网站中充满了冗长的新闻稿、自我推销的公司信息，以及几乎没什么用、根本就没人阅读的内容。

即使你认为你的产品 / 想法 / 项目是世界上最棒的，也不意味着有人愿意读长篇大论来了解它，甚至那些通过访问你的网站已表现出一些兴趣的人们也不会。

用户浏览你的网站是为了找到他们想要的信息。你要确保没有任何事物阻碍他们。

1. 浏览

用户是以浏览的方式阅读网页的。即使网站的内容非常棒，用户对它的关注也不可能如网站拥有者那样密切。

他们走马观花式地浏览内容，这看一眼，那瞅一下。他们只会粗略地阅读页面，眼睛在链接、标题、图像和小块文本上短暂地停留。

数百万的网页在与你争夺用户的关注，所以不要浪费用户的时间，要让他们尽可能容易地获取所需信息。

2. 倒金字塔

使用户能以最快的速度获取所需信息的关键之一，是使用一种新闻学上称为“倒金字塔”（interved pyramid）的编写方式。

回退到报纸还是印刷在真实的纸张上而不是在网站上发行的时候，编辑每天不得不面对的问题是如何规划物理版面的大小。他们必须生产出确切数量的内容来填充版面，不能多也不能少。但在新闻还没有发生之前，想要提前计划出新闻所需的版面几乎是不可能的。

所以当记者为报纸撰写文章时，他们有确切的字数限制，并且也明白，如果有更重要的新闻发生，他们可能必须缩短故事以腾出版面。

为了使这样的过程更容易，记者将他们故事最重要的细节放在头几段，并在随后的段落中包含那些相对不太重要的细节以及背景信息。这样，如果编辑需要缩短故事以减小占用的版面，就可以砍掉结尾的那些段落而不会丢失重点。图 2-1 能让你更好地理解这种工作方式。

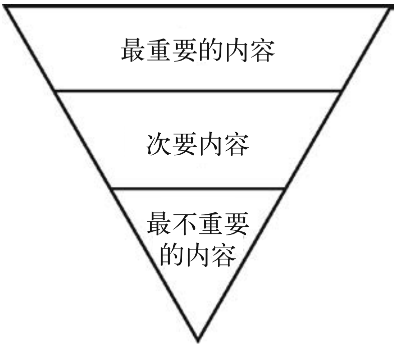


图 2-1：使用倒金字塔方式，确保用户看到最重要的那部分内容

撰写用于网站的内容时，你也可以使用相同的办法，不是因为编辑会截断你的故事结尾，而是用户可能会这么做，他们不会从头读到尾。确保最重要的细节首先出现，然后是用户可能看也可能不看的次要细节。

3. 分级标题

如果你的页面内容超过两三个段落那么长，把它分割成小块对大家肯定都是有好处的。

对用户而言，使用分级标题的好处有：

- 在开始阅读之前，更容易地了解页面结构，更好地认知页面内容的类型；
- 可以向前跳转到感兴趣的部分；

- 使用辅助技术的用户（比如使用屏幕阅读器软件的盲人用户）可以在页面内容中导航而不必通读所有内容；
- 可以在滚动时更容易地找到自己所在的位置。

对网站所有者而言，使用分级标题的好处有：

- 让搜索引擎知道页面的关键主题是什么，进而能够提升页面的搜索引擎排位；
- 内容作者能够创作出更适合 Web 的内容。

要写出好的标题，首先应将你的内容根据不同的重点分割成多个部分。

标题应该尽可能短并且表明重点，还应包含相关的关键词。理想情况下，每个标题应该以关键字开头，以使略读更容易。

标题应该是接下来内容的一个描述性预览，同时不应故弄玄虚。

4. 简短明了

用户在浏览页面的时候，倾向于阅读小段的文字内容。大块的文字会让用户望而生畏，通常不再继续阅读下去。

段落应该简短，一般不超过 100 字。

如果内容更适合按要点列出，那就应该使用列表。对于同样的内容，列表形式要比段落形式更易于查看和理解。

2.3.2 简明的语言

你在内容中所使用的语句十分重要，它不仅奠定了你的内容基调，也决定了用户是否能够理解内容。

根据研究，如美国国家教育统计中心（NCES）资助的“2003 年全国成人读写能力评估”（NAAL，http://nces.ed.gov/NAAL/），美国成年人的平均阅读水平通常在八九年级水平，近 50% 的成年人的阅读能力在六年级水平以下。

我们这些为网站撰写和编辑内容的人通常受过良好的教育，而我们的阅读水平往往也反映出这一点。但要记住我们的读者并不是我们自己。

很容易做出这样错误的假设：你的网站的用户肯定比其他人更聪明或受教育程度更高。毕竟，他们足够明智地选择了你的网站，不是吗？但如果没有浅显易懂的语言，你将失去潜在客户或是遗漏了向人们传递信息——他们很可能真的需要这些信息。

近年来，涌现了一场*简明语言*（plain language）运动，内容作者努力创作容易理解的内容，并以一种使用户更易于找到其所需内容的逻辑结构来编排它们。

使用简明语言的一个附带好处是，它真的有助于所有用户。大多数人都想要迅速地得到他们想要的信息，别忘了，有整个互联网的信息等着他们去读。即使对于文化程度高的用户，简明的语言也将帮助他们更快地阅读和导航内容。

你也能帮助英语不是母语的用户。这并不仅仅是指那些外来移民，也指想要光顾你的餐厅的外国游客，或者想要订购你的产品的他国居民。如果你的网站提供了信息，有兴趣阅读的可能不仅仅是你自己国家的居民。

对于使用简明语言，一个好的开始是用尽可能简单的词语，尽可能短的句子和段落。通过像前面所谈的那样精减你的内容，确保低文化程度的用户不必花费很多额外的时间来吃力地阅读与他们无关的内容。

以下资源可以帮助你了解简明语言：

- “The Center for Plain Language’s Plain Language Checklist”（http://centerforplainlanguage.org/5-steps-to-plain-language/）；
- “Lower-Literacy Users: Writing for a Broad Consumer Audience”，Jakob Nielsen 著，2005 年 3 月 14 日发表于 Nielsen Norman Group 网站（http://www.nngroup.com/articles/writing-for-lower-literacy-users/)；
- “United States Federal Plain Language Guidelines”（http://www.plainlanguage.gov/howto/guidelines/FederalPLGuidelines/TOC.cfm）。

2.4 内容平等

在思考自己对用户了解多少的时候，你需要考虑的第一件事是，你所“知道”的有关他们的大多数事情都是基于假设，不一定正确。

一个常见的错误是基于用户所使用的设备去假定用户需要或不需要某些内容。

例如，经常设想手机用户是“在移动中”，因此他们只需要基于位置的信息，比如要去的餐厅的地址，或者他们的航班是否准时。

但是现在手机已经变得无处不在，我们几乎在任何地方都使用它们，即使我们旁边就有笔记本电脑或台式机。

手机现在就像是另一台电脑，人们期望无论是在手机上还是在台式机或笔记本上浏览网站，都能获得同样多的信息。

每个人都应能访问网站的所有内容，不管他们正在使用什么设备。这一概念称为“内容平等”（context parity）。

响应式设计的优点之一，就是我们可以为所有用户提供相同的内容而不用顾虑他们所使用的设备，与此同时，我们提供的内容又为用户所使用的设备进行了优化。

正如你将在第 5 章中学到的，要针对不同设备的屏幕尺寸呈现不同的内容，你有很多事可以做。

2.5 内容管理

如果你的网站有内容（所有网站都有内容），你需要制订一个*内容管理* (content governance）计划。这意味着一旦内容在你的网站上发布，你就要妥善管理它们。

你不能发布完内容就将其抛诸脑后，否则会出现信息过时或者不正确、链接失效等问题。在网站上发布内容之前，你需要为内容管理制订一个良好的计划，否则你可能永远不会再有时间去做这件事。

创建长期有效的内容

要使你的内容更易于维护，你能做的一件事就是使其尽可能不受时间的影响。

当然，不是所有的内容都是永恒的，做好这件事的一个方法是审视内容并想象你在一年以后又碰巧遇到它。那时它还能不能独立存在？

尽可能在日期中包含年份。你可曾有过这样的经历：你听说了一个年度盛事，想要参加，于是到网站上查看，结果看到上面醒目地写着盛事于 10 月 15 日举行，可你却不知道这指的是今年即将举行的活动，还是去年已举办过的活动？

如果你知道信息仅在特定时间内是有效的，应该在内容中做个标注。这样如果你忘记回过头去修改或删除，至少看到它的人会知道内容已经失效。

2.6 自适应内容

作为响应式网站，网站中的内容不仅要能在不同宽度的屏幕中浏览，还应该能在不同的场景中完整地浏览。

例如，图 2-2 是 Rachel Nabors 发表在 *A List Apart* 上的一篇文章在桌面浏览器中的显示效果。



图 2-2：在桌面显示器上浏览 *A List Apart* 上的一篇文章

但有些人喜欢用 RSS 阅读器阅读帖子和文章。阅读器会剥除网站的所有设计，只依靠 HTML 来显示内容，如图 2-3 所示。

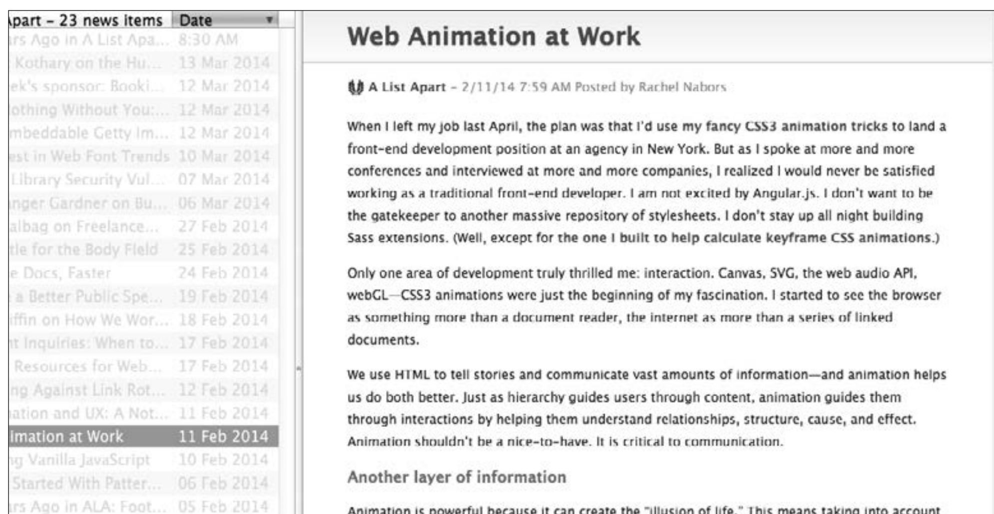


图 2-3：在 RSS 阅读器中查看一篇文章

人们也常常使用 Instapaper（http://www.instapaper.com/）之类的服务，用它们查看网页的主要内容（广告或导航之类的东西通通没有），或者保存文章以便稍后阅读。图 2-4 显示的

是台式电脑上的浏览效果，图 2-5 显示的是 iPhone 上的浏览效果。



图 2-4：在台式电脑上查看 Instapaper 中的一篇文章



图 2-5：在 iPhone 上查看 Instapaper 中的一篇文章

我们不能再仅仅考虑如何在网站上显示内容。我们需要把它看作一系列的数据块，比如标题、作者姓名和主体内容。这些不同的数据块称为*元数据*（metadata），意思就是“关于数据的数据”。

如果你使用 CMS，那你可能一直是这么做的，用单独的字段对应标题、文章日期和作者姓名。在响应式设计中，我们依据屏幕尺寸来调整布局。在小屏幕上，标题、日期、作者和主体文本可能会一个接一个地垂直显示。在大屏幕上，日期和作者则可能是在另外一个文

本框中显示。这些布局的变化仅在内容被分割成单独的数据块时才有可能实现。

如果文章重现于另一个应用中，数据块也使得内容能以一种合理的方式显示。

那如何处理其他类型的内容呢？例如，在一个电子商务网站中，不能只用一个描述字段来包含销售商品的所有细节，而应使用单独的字段分别描述颜色、尺寸、材质、使用类别等。这样，你不仅可以根据屏幕宽度轻松地更改页面布局，还可以实现更强大的搜索和导航功能，让用户更快地找到所需的商品。

通过结构化内容并附加元数据，你不仅能使内容更好地适应不同的屏幕尺寸，还能使内容更易于显示在网站之外的地方。

2.7 总结

不久之前，网站内容仍被视为在设计之后再添加的东西，但近年来内容策略已显得愈发重要。

在设计之前就开始考虑内容将帮助你创建一个符合商业目标和用户需求的网站。

在向新站点转移的过程中要整理你的网站内容，首先通过内容审计来查看你拥有的内容，然后进行梳理和精减，只使用网站真正需要的内容。不必要的内容只会使用户难以使用网站，使网站所有者难以维护网站。

确保内容简洁，并以简明的语言编写，这样易于用户阅读。用户是以浏览的方式阅读网页的，所以要创建短的段落并用分级标题将内容分成多个部分。确保最重要的信息放在页面开头，因为有时用户不会从头读到尾。

不要基于用户访问网站所用的设备来对其所需的内容做出假设。所有来到你的网站的用户都应该能访问全部的内容。

除了添加内容到新设计的网站之外，还要考虑在网站上线后如何管理内容。要制订更新内容的计划。尽可能创建不受时间影响、长期有效的内容。

将内容分块使之可在页面上重排，也很容易移植到其他媒体上。

在下一章，我们将开始学习响应式网站背后的代码细节。

第二部分

创建响应式网站

第3章

响应式网站之HTML

创建一个响应式网站，你要用到 HTML 和 CSS，偶尔也会用上一点儿 JavaScript。

即使你的工作并不涉及为网站创建代码，也应该知道一点儿发生在响应式网站“底层”的事情，所以请不要跳过接下来的几章。

本章假设你至少对 HTML 有一些基本的了解。如果你不是程序员，那么不必完全精确地理解一切是如何工作的，只需做一般性的了解。

即使你有丰富的 HTML 编码经验，本章中所涉及的概念可能仍有你不熟悉的，比如视口（viewport）属性以及使用 HTML5 中新的结构性元素来创建网页。

对于其他概念，如文档类型（doctype）和字符集（charset），你可能并不陌生，但在这里讲述它们，是因为它们对于理解如何正确地创建一个响应式网站的 HTML 页面是必需的。

在本章中，我们也会讨论一下 HTML 的版本，以及使用 HTML5 创建网站的意义。然后，我们将使用 HTML5 的结构性元素创建一个示例网页。

在本章的结尾，我们会讲讲如何写出好的代码，包括使用语义化 HTML、分离内容和表现，以及养成使用 HTML 元素正确标记内容的良好习惯，这能让所有用户更易于理解它们。

[小贴士]

如果你完全不懂 HTML，Mozilla 开发者网络中的“HTML 入门”是一个开始学习它的好地方（https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Introduction）。

3.1 使用HTML

HTML 不是一成不变的：它随时间不断地修订以适应新技术的发展并加入新的特性。HTML 是一个活的有机体，它会随着时间改变自身以适应其背后网络和技术需求的不断发展。

每隔几年，就会有一个 HTML 新版本发布。在本书出版之时，浏览器仍处在 HTML4 切换至 HTML5 的过程中。

本书中的例子将使用最新版本的 HTML5。但因为 HTML5 还处在一个发展的过程中，并非所有的 HTML5 新元素现在都是可用的。在本节，你将了解 HTML 版本是如何制定的，以及如何找出那些可以在网站中安全使用的 HTML5 新元素。

3.1.1 HTML版本

很多 HTML 元素在版本的变更中保持不变，所以每次新的 HTML 版本出来后你并不需要从头开始学起。

打一个容易理解的比方，如果你阅读莎士比亚写于 500 年前的某些作品。它们是用英语写的，与我们今天讲的是同一种语言，但又有些不同。在莎士比亚的作品中，一些单词在今天的英语中已经不再使用。其他一些仍在使用的单词，也可能拼写不同或含义有细微差别。同样，如果莎士比亚突然穿越到现代，有很多单词他也不会认识。

但即使有这些差异，它仍然是同一种语言，你也差不多能完全理解它。

在比较不同版本的 HTML 时也是如此。HTML 由*元素*（element）组成，其是用于描述网页内容的组件（component）。例如，段落元素用于表示段落，图像元素用于向网页中添加图像。

每一个新版本的 HTML 中都会加入新的元素，某些已有的元素会有稍微不同的含义，某些元素则将消失不见。比如，段落元素 `<p>`，自 HTML 的第一个版本起含义一直未变，但像 `<key>` 元素则已经不存在了，画布元素 `<canvas>` 是最近才添加至新版中的。

在大多数情况下浏览器能够识别任一版本的 HTML，但老的浏览器可能无法识别某些新元素，所以你需要做一点额外的工作来确保它们可以正常地显示全部内容。

与人类语言一样，HTML 的演变也经历了一段时间，它并非一步到位。新版的 HTML 规范提交后，Web 社区会花费大量时间（以年计）一点一点讨论它。然后浏览器开始一点一点实现它。就在新 HTML 规范中的一部分已先被浏览器实现时，剩余部分仍在讨论和测试中。

各版本之间存在着很多的时间上的重叠。比如 HTML5 的制定工作始于 2004 年，最终的规范是计划于 2014 年末“完成”（现在它还只是一个“草案”）。而“完成”并不意味着改变

就此终结，接着将会有 5.1、5.2 这样具有较小变化的版本。下一个大版本 HTML6 也已经在讨论了，而老版本的 HTML 仍在使用中。

除了较新的网页是以 HTML5 创建的外，你还会遇到大量以 HTML4 和 XHTML（出现于 HTML4 和 HTML5 之间）创建的网站。

3.1.2　Web标准

万维网联盟（W3C，World Wide Web Consortium）是一个自愿性的标准化组织，对“官方”版本的 HTML 具有决定权。（*自愿*意味着尽管行业已同意使用 W3C 制定的规则，但并非被要求要这么做。）

在 W3C 出现之前，浏览器已建立起自己的规则来解释 HTML。因此，一个网页在不同的浏览器上的表现可能会非常的不同，能在一个浏览器中工作的 HTML 代码在另一个浏览器中甚至可能不被识别。

因为 W3C 是自愿性质的，浏览器实际上并不是必须要遵守它的规则。但在大多数情况下，浏览器会遵守它们，因为如果一个浏览器显示的网页效果与其他浏览器大相径庭，那不会有人想要使用它。可还是会存在某些差异，所以接下来你将学习在浏览器不遵守规则的情况下，需要在代码中做什么才能确保网页被正确地显示。

作为网页设计师，我们需要遵守这些规则。这将确保我们的网站能够在不同的浏览器和设备之上工作。遵循标准也将确保你的网站是未来友好型的，更有可能与将来出现的新设备及新技术兼容。

3.1.3　使用HTML5

当人们谈及一个“HTML5 元素”时，通常是指一个 HTML5 中新出现的元素（也就是说，不是 HTML4 中的元素）。

如果一个元素是新加入到 HTML5 中的，你需要了解浏览器是否已经支持该元素。即使它们被支持，你也要知道某些用户的电脑上仍使用的是老式浏览器，并不支持某些新元素。

当我在本书中引用新的 HTML5 元素时，我会告之它们目前是否被良好支持，以及你需要怎么做才能使之适应老式浏览器。

正如我前面提到的，大部分 HTML 元素在各个版本中是保持不变的。

HTML5 中有超过 100 个不同的元素。在本书中，我们只学习那些你可能会用到的元素。但还有很多其他的元素，如果你遇到一个不认识的 HTML 元素，查一查它的相关资料以确定它的作用以及它是否是有效的 HTML5 元素。

[小贴士]
需要一份所有版本 HTML 元素的完整列表，可查看 Mozilla 开发者网络上的“ HTML element reference”（ https://developer.mozilla.org/en-US/docs/HTML/Element ）。

要想弄清楚哪些新元素是被哪些浏览器支持的，参考 Can I Use 网站（ http://www.caniuse.com ）。

3.2　页面基本结构

从头创建一个网页所要做的第一件事是以基本的 HTML 元素来构建页面。

3.2.1　文档类型声明

在 HTML 文件开头要做的第一件事是*文档类型声明*（doctype）。文档类型声明告诉浏览器，页面使用的是哪个版本的 HTML。

对于 HTML5，文档类型声明是：

```
<!DOCTYPE html>
```

如果你现在所工作的网站使用的旧版本的 HTML，你会看到一些其他的文档类型声明。

即使在一个新的 HTML 版本出来之后，使用旧版本 HTML 的页面仍能继续工作，但它们必须有一个正确的文档类型去匹配所使用的 HTML 版本，否则浏览器可能无法显示该页面。

如果声明的文档类型与页面使用的 HTML 版本不匹配，浏览器将做某些猜测，这也许能让页面正常显示，但不会总是如此。所以不要指望靠浏览器猜测来工作。如果网页非如你所想的那样显示，请检查并确保你的页面拥有正确的文档类型声明（与你所使用的 HTML 版本匹配）。

[小贴士]
在 W3Schools 的“HTML <!DOCTYPE> Declaration”（“HTML <!DOCTYPE> 声明”，http://www.w3schools.com/tags/tag_doctype.asp ）中你可以了解先前那些文档类型之间的差别。

3.2.2　文档结构

文档类型声明之后的所有内容，构成网页的 HTML 代码，其进一步被明确分为两部分，即 `<head>` 元素部分和 `<body>` 元素部分：

```
<!DOCTYPE html>
<html>
<head>
```

```
...
</head>
<body>
...
</body>
</html>
```

请注意，当你在代码示例中看到“…”时，这表示其他代码或内容将出现在那里，“…”不是代码的一部分。

你会发现我们在这里使用的几个元素都是以成对的方式出现的，如 `<head>` 和 `</head>`。大多数 HTML 元素（并不是全部）用开始 / 结束标记来包含它们描述的内容。这些开始 / 结束标记间的所有内容构成了页面的 `<head>` 元素部分。

`<html>` 元素应该总是有一个 `lang` 属性，告诉浏览器页面内容所使用的语言，在这里，`en` 表示英语。尽管这个属性是可选的，但它很重要，因为它可以告诉屏幕阅读器以何种语言来读入，也可以帮助浏览器决定如何用连字符连接单词。

> **[小贴士]**
>
> 如果你的网站使用的是英语之外的其他语言（尤其是使用不同字符集的语言），或者是多语言的，要确保你的网站能正常工作，还有几件事是你必须知道的，可以先看一下 W3C 上的“Internationalization Techniques: Authoring HTML & CSS”（http://www.w3.org/International/techniques/authoring-html-dynamic）。

`<head>` 元素包含页面的相关信息（通常称为*元数据*，使用 `<meta>` 元素表述）。它还可以包含外部 CSS 或 JavaScript 文件的链接，以及嵌入式样式或脚本。在 `<head>` 元素中的内容不会像页面内容那样显示在页面上。

`<body>` 元素包含页面的所有内容。

3.2.3　页面标题

页面的 `<title>` 元素的内容并不会出现在网页中，而且实际上也可以与你在页面中看到的标题完全不同。

不过，你写在 `<title>` 元素中的语句应该是经过深思熟虑的，因为它在其他的很多地方是被用作页面标题的，比如：

- 在浏览器窗口的顶部，或页面标签上（具有多标签浏览功能的浏览器）；
- 在页面被收藏后，用户的收藏栏中；
- 在搜索引擎结果指向页面的链接中。

通常，<title> 元素的内容由站点名和页面标题组合而成，并用破折号、冒号或管道符号分隔。

举几个例子：

```
<title>Metro - The Boston Globe</title>
<title>Flickr: Groups</title>
<title>Ways to Help | American Red Cross</title>
```

尽管 <title> 元素的内容可以不同于出现在页面上的标题，但它们应该是大致相似的，否则可能会使用户感到困惑。

长标题在搜索引擎结果中会被截断，所以应控制标题的长度。通常的指导方针是不超过 64 个字符，尽管每种搜索引擎对标题最大长度的限制不尽相同。

3.2.4 字符集

字符集（charset）规定了你 HTML 文档（你在 HTML 文件中输入的所有 HTML 代码和内容）中字符的编码。

字符集不是 HTML 元素，它是 <meta> 元素的一个属性，在 HTML 文档的 <head> 元素中你可以有一个或多个 <meta> 元素，它们用于提供页面的相关信息。

字符编码本质上是将字母、数值和符号翻译成计算机能够识别的二进制语言的代码。

你可能没有意识到，无论你用何种应用程序创建你的 HTML 文件（比如文本编辑器），都会有一个默认的编码。每种浏览器也都有一个默认的编码。如果它们所使用的编码不同，有些内容会在转换过程中丢失，所以你需要告诉浏览器使用的是什么编码，否则页面可能无法显示。

对于 HTML5 页面你应该使用 UTF-8 字符集，这是网页中使用最广泛和最灵活的字符编码。它支持大多数国际语言中的字符：

```
<meta charset="utf-8">
```

旧版本的 HTML 声明字符编码的方式有所不同。如果你现在所工作的网站使用的是旧版 HTML，你可以参考一下 W3C 网站上的“在 HTML 中声明字符编码”（http://www.w3.org/International/questions/qa-html-encodingdeclarations）。

[小贴士]

有时当你从一个应用程序复制粘贴文本到一个纯文本编辑器中，一些符号（如“智能”引号）将被问号或小方框取代。这意味着两个应用程序使用的是不同的编码。你需要重新输入这些字符（或者做下搜索与替换）。

3.3 视口

视口（viewport）元数据元素在大多数非响应式网站中通常是不使用的，但却是使你的响应式站点工作的关键。

> [小贴士]
>
> 如字符集一样，viewport 也是 <meta> 元素的一个属性。你所提供的每一种类型的元数据必须在放在单个 <meta> 元素中，而不能简单地在一个元素中组合所有的属性。

视口是计算机或设备屏幕上你浏览网页的那块区域。在第 1 章中，你已经知道响应式设计中的媒体查询是基于视口宽度的。

在台式电脑上，如果你启动浏览器窗口，去掉菜单、工具栏、滚动条及浏览器自身的其他部件，剩下的部分就是视口了，如图 3-1 所示。



图 3-1：虚线所示空间即是视口区域

在移动设备上，视口的宽度就是屏幕的宽度。

viewport 这一元数据属性会指示浏览器将以怎样的大小来显示网页才能使其在视口中正确地显示。

未设置 viewport 属性的网页在桌面显示器上是以全尺寸方式渲染（或显示），这是你习惯的浏览方式。

默认情况下，手机浏览器渲染一个没有设置 viewport 属性的网页时，会先像桌面浏览器那样渲染页面，然后按比例缩小它来适配手机视口，如图 3-2 所示。如果有媒体查询是基于视口的宽度，那它们不会生效，因为浏览器是按桌面显示器尺寸来计算视口宽度的。



图 3-2：威斯康辛大学的整个网站是以桌面尺寸渲染的，迫使用户放大文字来阅读

大多数手机浏览器会使用 980 像素这一默认视口宽度渲染一个网页。想象一下你有一个 980 像素宽的桌面显示器，你在其屏幕上看到的内容将按比例缩小以适应手机的屏幕。如果网页比 980 像素宽，它可能还得进一步缩小以适应屏幕。

对于响应式网站，我们不希望手机浏览器渲染页面如同它是显示在 980 像素宽的桌面显示器一样，我们希望它按移动设备真实的屏幕尺寸（可以是任何宽度）来渲染页面，因为我们的网页已针对小屏幕宽度进行了适应性设计，并不需要缩放。

在图 3-3 中，我们可以看到《波士顿环球报》的首页在 iPhone 屏幕上的显示效果，而图 3-4 中，则是其在 iPad 屏幕上的显示效果。这个网站在两个屏幕上都是全尺寸渲染的，尽管它们的物理宽度不同。



图 3-3：《波士顿环球报》网站以全尺寸显示在 iPhone 上



图 3-4：《波士顿环球报》网站以全尺寸显示在 iPad 上

一般来说，对于响应式站点，应设置视口如下：

```
<meta name="viewport" content="width=device-width,
initial-scale=1">
```

3.3.1 宽度

属性 `width` 告诉浏览器如何按比例缩放网页。

对于响应式网站，值 `width=device-width` 告诉浏览器以全尺寸渲染页面（不管屏幕尺寸可能是多少）。因此在浏览器渲染页面时，视口宽度就是设备的实际宽度（即 100%）。

如果没有在 `viewport` 元数据属性中指定设备的宽度，设备将使用它自己默认的视口设置来渲染页面。比如，iPhone 的默认宽度是 980 像素，所以在图 3-2 中，浏览器渲染页面如同浏览器窗口是 980 像素宽一样，但接着缩小它至 1/3 的尺寸，以便它适应只有 320 像素宽的屏幕。

如果你使用了 `width=device-width`，当用户变换设备方向时，用来渲染页面的宽度将自动调整。因此，如果用户转换她的 iPhone 方向，从竖向变成横向，以 320 像素宽渲染的网页将自动以 480 像素宽（你水平握持手机时的屏幕宽度）重新渲染。

3.3.2 初始缩放比

属性 `initial-scale` 告诉浏览器在屏幕上首次加载网页的时候如何缩放它（即缩放因子）。

使用值 `initial-scale=1` 意味着页面将按 `width` 属性确定的宽度来渲染，不放大也不缩小。

如果你用的是大于或小于 1 的数值，则页面将缩放至对应的比例。例如，值 `initial-scale=2` 意味着页面会放大至实际大小的两倍，因此你只能在屏幕上看到一半的页面。

> [小贴士]
>
> 如果你工作于一个根本不包含 initialscale 属性的非响应式网站，这将导致你的网站在移动设备上被放大到一个巨大的尺寸，使得用户无法一次看到整个页面。

3.3.3 用户可缩放

值 `initial-scale` 仅决定网页在屏幕上首次加载时的大小。但要记得，你的移动设备还给予了用户放大或缩小内容的能力。

以下 `viewport` 值将关闭用户缩放内容的能力：

```
user-scalable=no
```

不过，对于大多数网站，你应该避免使用这个值，让用户在需要时可以进行缩放。尽管响应式站点不管屏幕的宽度是多少都会以全尺寸加载，但也许仍存在用户可能希望放大文字的情况，比如他们阅读小文本有困难的时候。

你应该只在非常有限的情况下关闭缩放，比如用 HTML/CSS 创建的游戏或其他的一些 Web 应用程序。

如果你不包含这个值，那默认值是 `yes`，其允许用户对页面进行缩放。

3.3.4　最大缩放比

你也可以使用 `width` 属性来设置页面的 `maximum-scale` 值。例如，`maximum-scale=2` 意味着你的用户最多只能按比例放大页面至全尺寸的两倍大小，之后就不能再放大了。

设置 `maximum-scale=1` 意味着用户完全不能放大，与使用 `user-scalable = 1` 具有同样的效果。

再重复一下，你应该避免使用此值，除了在非常特殊的情况下。

> [小贴士]
>
> 你可能已经注意到，我在书写元素名和属性名时都用的是小写，在 HTML5 中，元素和属性是不区分大小写的。也就是说，你可以用小写字母、大写字母或任意组合来书写元素和属性。大多数开发人员使用小写字母，因为小写形式的单词比大写形式更容易阅读。

3.4　结构性元素

`<body>` 元素包含所有的页面内容。

首先，我们将使用 HTML5 结构性元素将页面划分为几个主要的部分，比如 `<header>`、`<nav>`、`<footer>`（它们称为结构性元素，因为它们定义了页面的结构）。这使得我们更容易将样式应用于页面的不同部分。

每个页面部分的元素都有描述性的名字，如 `<header>`、`<footer>`、`<nav>`（导航）、`<section>` 和 `<aside>`。

这些元素是 HTML5 中的新元素。此前，网页设计师使用通用元素如 `<div>`（区块）将页面分成几个部分，然后使用 `id` 或 `class` 属性来标识每个元素，以便在应用 CSS 时能针对特定的部分。

使用结构性元素是标记语义化的一个例子，标记语义化是指使用能准确地描述页面内容含意的元素，而不是像 <div> 那样的通用元素。

[小贴士]

虽然我向你展示了如何使用特定的元素作为首选方法来构建 HTML 页面，但它们并不是必需的。事实上，没有元素必须包含在网页的元素中。

3.4.1　屏幕阅读器

此外，结构性元素还能帮助用户方便地使用辅助技术来导航页面，这些辅助技术包括能够为盲人用户朗读页面内容的屏幕阅读器，或者语音命令软件，它允许用户在无法使用键盘或鼠标的情况下与计算机交谈（例如，用户可能会说“导航”）。

[小贴士]

在创建一个网站时，你需要确保网站对于残障人士的可访问性。我们将在第 8 章中深入探讨可访问性。Web 可访问性与可用性密切相关，你可以查看 W3C 中的文章“Web Accessibility and Usability Working Together”（http://www.w3.org/WAI/intro/usable）来做更多的了解。

但是，并非所有的老式浏览器和屏幕阅读器都能识别这些新元素，所以我们需要做一些额外的努力来照顾它们。

在这些新元素出现之前，已有一个单独的规范来帮助网站使之是完全可访问的。这就是同样出自 W3C（为我们制定了 HTML 规范）的 WAI-ARIA（Web 可访问性倡议——可访问的丰富互联网应用）。它是一组可以被添加到 HTML 元素中的属性，用以提供额外的信息和功能。

WAI-ARIA 中的一部分是角色（role）属性，它们可以在 HTML 元素中设置，以提供更多关于元素作用的信息，许多角色属性可以与新的 HTML5 结构性元素对应。

新的结构性元素确实棒极了，但因为它们是刚刚新加入到 HTML 中的，所以不是所有的浏览器和屏幕阅读器都能够利用这些元素，我们还需要在我们的结构性元素上包含相应的 WAI-ARIA 角色属性。

新的浏览器会查看 HTML 元素，而老式浏览器则查看 WAI-ARIA 角色属性。

我将在讲解每个页面元素时同时对 WAI-ARIA 角色属性进行解释。虽然它们不是必需的，但最好在你的代码中包含它们，以使网站拥有最大限度的可访问性。

3.4.2 <header>

通常，在页面开头，<body> 元素中的首个子元素是 <header>。(注意 <head> 和 <header> 是不同的元素，它们容易让人混淆。)

依照 HTML 规范，在 <header> 元素中应包含介绍性和导航性的辅助内容，这可能包括 Logo、搜索功能和主导航栏。

除了整个页面可以有个 <header> 元素，你也可以在页面的其他地方增加 <header> 元素，用来包含针对特定页面区块的介绍性和导航性的辅助内容。

对应的 WAI-ARIA 属性是 banner，因此你只需将其作为一个属性添加到 <header> 元素：

```
<header role="banner">
...
</header>
```

3.4.3 <nav>

页面主导航包含在一个 <nav> 元素中，与页面中的二级导航一样。通常在 <header> 元素中会有一个 <nav> 元素，但也并非总是如此。

导航可以是连接站点其他页面的链接，也可以链接同一页面上的内容，比如在一个目录表中。

一个页面中可以有多个 <nav> 元素。相应的 WAI-ARIA 属性是 navigation：

```
<nav role="navigation">
...
</nav>
```

3.4.4 <footer>

脚标元素 <footer> 是页面上的一个区块，不一定在最后，其包含页面内容的有关信息。

例如，一个脚标可能包含版权声明，页面相关信息，以及连接到相关资料（比如隐私政策或联系页面）的链接。

除了用作整个页面的页脚，你也可以在页面的其他地方设置脚标，对内容区块而不是整个页面执行相同的功能。

在一个页面中可以有多个 <footer>。相应的 WAI-ARIA 属性是 contentinfo：

```
<footer role="contentinfo">
...
</footer>
```

3.4.5 <article>

一个 <article> 元素是网页上一块独立的内容。它可能是一篇报纸文章，一篇博客或论坛帖子。因为它是独立的，这种类型的内容可以通过内容聚合来分发（如 RSS 订阅）。

一个页面可以有多个 <article> 元素。没有 WAI-ARIA 属性对应于 <article> 元素：

```
<article>
...
</article>
```

3.4.6 <aside>

一个 <aside> 元素用于次要的内容，可以是与网站或整个页面相关的（例如一个友情链接），或者如果它是嵌套在一个 <article> 元素中，那就是与文章相关的（例如一个术语表）。

你可以在一个页面上拥有多个 <aside> 元素。相应的 WAI-ARIA 属性是 complementary：

```
<aside role="complementary">
...
</aside>
```

3.4.7 IE支持

对于这些新的 HTML5 结构性元素，有一个小问题就是一些老的浏览器不能识别这些元素。

对于大多数浏览器，这没什么大不了的。它们对待无法识别的元素就像对待普通的 <span> 元素一样。

但是，尽管旧版本的 Internet Explorer（IE8 及其更早版本）会像 <span> 元素一样来显示这些元素，但却也会拒绝对它们应用任何 CSS 样式，这可真的会搞砸一个页面的设计。

所以，如果你想让你的网站能够在那些老版本的 IE 中正确显示，你需要加一点 JavaScript，这会有助于浏览器识别这些新元素。

有两种方法来做这件事。而且不用担心，使用它们并不需要你懂 JavaScript！

第一种变通方案是使用所谓的腻子（polyfill）或利刃（shiv）脚本，其是一小段用来使一个老式浏览器变得像新浏览器一样的代码。

HTML5 Shiv（https://github.com/aFarkas/html5shiv）是由几位开发人员（见 GitHub 页面中所有贡献人列表）创建和改进的一段代码，你需要做的所有事情就是按照页面上的说明，下载一个 JavaScript 文件，并将它添加到你的网站中，然后在站点页面的 <head> 元素中链接它（如果你不知道 Bower 是什么，使用手工安装说明）。

安装说明指示我们使用以下代码来添加 shiv 到我们的网站：

```
<!--[if lt IE 9]>
    <script src="files/html5shiv.js"></script>
<![endif]-->
```

第一行代码表示仅在浏览器是 Internet Explorer 且版本早于 IE9 时才运行脚本。

也还有其他的资源，比如：Modernizr（http://modernizr.com/），它做与 HTML5 Shiv 相同的事情。

3.5 创建一个页面

对于我们的示例页面，我们将从一些常见的元素着手。

我们将创建一个有关大熊猫的网站，标题是"永远的大熊猫"，为简化例子，我们只使用一些基本的内容片段。

在大多数的网站上，你会使用 CMS（包括如 WordPress 那样的博客软件），所以 HTML 代码可能出于编辑目的分布于数个单独的文件，而不是全都在一个页面上，但对于本例的目的而言，我们的 HTML 代码都在一页上。

如果你是网页制作的新手，先在你的电脑上打开一个文本编辑器，如 Mac OS 中的 TextEdit，或 Microsoft Windows 中的记事本。以".html"扩展名，而不是".txt"，保存包含你的 HTML 代码的文件。然后，你就可以在浏览器中打开并浏览它了，就像网页那样。

3.5.1 结构性元素

在 <body> 元素中，首先以 <header> 元素开始，它包含了页面标题，紧接着是 <nav> 元素，它包含了导航链接。

在 <header> 元素之后是 <article> 元素，我们会有一篇博客内容包含在其中。然后用 <aside> 元素创建相关链接部分，因为它是页面的补充信息。

页面将以 <footer> 元素结束，它包含了一些附加信息，比如我们的版权声明：

```
<body>
    <header role=""banner">
        <nav role="navigation">...</nav>
    </header>
    <article>...</article>
    <aside role="complementary">...</aside>
    <footer role="contentinfo">...</footer>
</body>
```

[小贴士]

你会在 HTML 页面中经常看到元素以不同的层次缩进，就像上面的例子那样。这不是必须的，浏览器会忽略掉额外的空格，但它有助于帮助开发人员看清页面结构，并能够更容易地匹配开始和结束标记。

3.5.2　加入内容

请记住这一点：我们编写的页面必要时不借助 CSS 也能浏览。如果没有页面布局的话，所有内容应该按照我们想要它出现在页面上的顺序来编码，也就是一块内容接着另一块内容。

将所有的片段拼到一起，并加入内容，我们得到以下代码：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Pandas Forever</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,
    initial-scale=1">
    <!--[if lt IE 9]>
        <script src="files/html5shiv.js"></script>
    <![endif]-->
</head>
<body>
    <header role="banner">
        <h1>Pandas Forever</h1>
        <nav role="navigation">
        <ul>
        <li><a href="/">Home</a></li>
        <li><a href="/about/">About</a></li>
        <li><a href="/links/">Links</a></li>
        <li><a href="/contact/">Contact</a></li>
        </nav>
    </header>
    <article>
        <h2>Pandas in Wolong</h2>
        <p>The Wolong National Nature Preserve, in the
        Sichuan Province of China, is home to more than
        150 giant pandas. It's one of the key sites for
        panda breeding research, and 66 cubs have been
        born at Wolong since it was established in 1980.
        Pandas are an endangered species, with between
        1500 and 3000 living in the wild, and less than
        300 in captivity (research centers and zoos).</p>
    </article>
    <aside role="complementary">
        <h2>Related Links</h2>
```

```
            <ul>
            <li><a href="http://www.flickr.com/groups/
            pandasunlimited/">Pandas Unlimited</a></li>
            <li><a href="http://nationalzoo.si.edu/animals/
            webcams/giant-panda.cfm">National Zoo Panda
            Cams</a></li>
            <li><a href="http://worldwildlife.org/species/
            giant-panda">Panda Facts at WWF</a></li>
            </ul>
        </aside>
        <footer role="contentinfo">
            <p>&copy; 2014 Pandas Forever</p>
        </footer>
    </body>
    </html>
```

我们拥有所有先前提到的 `<header>` 中的元素，包括 HTML5 Shiv，我们上传 HTML5 Shiv 到网站的 files 目录中，并按安装指南说明的方式来链接它，以便 IE8 及其更早版本的浏览器能识别新的 HTML5 结构性元素。

网站标题使用 `<h1>`，页面标题使用 `<h2>`。导航设计的惯例通常是把它放在一个无序列表 `<ul>` 元素中，尽管不是必须要这么做。如果你愿意，对于每个导航项你也可以使用 `<span>` 或 `<div>` 元素。

在 `<body>` 元素中，我们添加了所有的内容。这是一个非常简单的页面，类似于你见过的文章或博客。

再往下，`<footer>` 元素包含了我们的版权声明。

3.5.3 无样式页面

首先从无样式的内容开始，图 3-5 是在没有应用任何 CSS 样式的情况下，页面在桌面浏览器中看到的效果，而图 3-6 则是在 iPhone 上看到的效果。

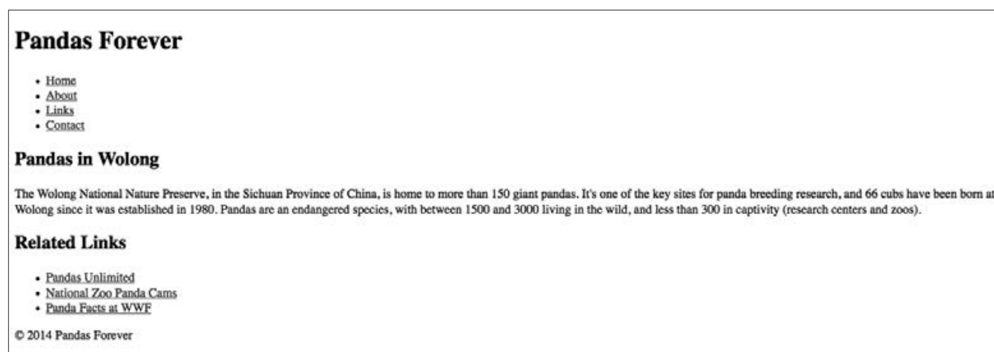


图 3-5：在没有应用 CSS 样式时，网站在桌面显示器上的显示效果

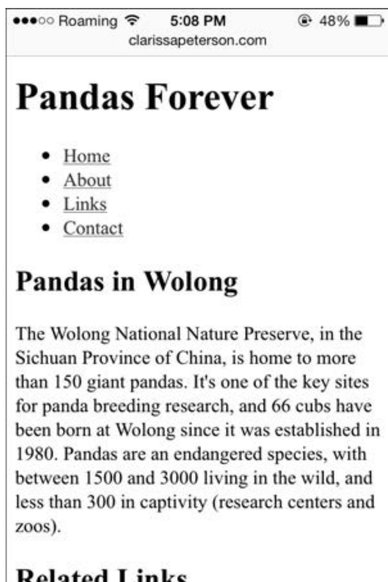


图 3-6：在没有应用 CSS 样式时，网站在 iPhone 上的显示效果

我们目前还没添加任何 CSS 样式，网站就已经是响应式的了！ HTML 默认是响应式的。

3.6　清晰和语义化的HTML

保持代码尽可能简洁是网站运行良好的关键。我并不是说要使设计过分简单，我是说当你创建代码来显示你的设计时，你要保持代码简单明了。

这将使你的站点代码更容易编写和维护。

正确地使用 HTML 元素，并确保你对所有元素都是熟悉的，那样你就知道什么元素符合你的需要。

一开始，你就应该确保把最基本的做好，比如对分级标题使用分级标题元素（且以正确的序号），对列表使用列表元素，等等。同时也要运用我们在本章早些时候所讲到的新元素 `<nav>` 和 `<section>` 去正确标注页面的各个部分。

`<div>` 和 `<span>` 是没有任何语义的容器元素。你可以在没有语义 HTML 元素能够表达你需要的含义时使用它们，但应谨慎，只在必要时为之。

3.6.1　分离内容与表现

HTML 和 CSS 是独立的语言，具有不同的用途。HTML 用于显示内容并提供内容的相关信息，并不提供任何样式或装饰。CSS 用于决定站点的视觉外观，如布局、字体、文字大

小和颜色。

你的内容可能并不总是以你所期望的样式来显示，所以确保不使用 CSS 来表达意义。

例如，使用适当的分级标题（<h1> 到 <h6>）将确保所有的设备和浏览器在显示内容时，即使不能应用 CSS，也知道哪一部分是最重要的。再比如，对于使用屏幕阅读器的盲人用户，如果阅读器软件知道页面是按分级标题分成若干章节，它就能让用户在章节间跳转，而不用从头到尾通朗读整个页面。

另一方面，如果你使用 CSS 来给标题一个粗大的字体，但是却忽略了使用恰当的 HTML 分级标题元素，浏览器实际上不会知道这些文字是描述页面上内容的标题。

即使排除了残障人士，仍会有很多人是从其他地方访问你网站的内容，而不是从你网站上的某个页面，因此他们看不到你的设计，只能看到你的内容。

这些用户可能使用的是我们在第 2 章中所谈到的 RSS 阅读器或稍后阅读（read-later）应用。为使他们能够正确地查看你的内容，内容必须是结构化的，这样软件才能够正确显示它们。

同时，一些最重要的访客——搜索引擎——来到你的网站时，它们看不到网站的设计。当搜索引擎软件（称为*机器人*）查看你的网站时，它只看 HTML，而不是设计。它使用 HTML 提供的信息来确定你的页面应该如何分类和出现在搜索结果中。

使用 HTML 结构化方法组织你的内容，这样搜索引擎就能够知道页面上不同部分的内容是什么，而不只是给它们一堆杂乱的文字。如果你的页面没有使用语义化代码，肯定会对你的网站在搜索引擎结果中的排位造成负面影响。

设计页面结构之初，首先是要确保你的内容在任何地方都可以被正确理解和显示。

3.6.2 注释

注释是你的 HTML（和 CSS）文件中的重要部分，尽管用户从不会看到它们。使用注释，你可以添加说明到源码中，浏览器在显示页面时会简单地忽略掉它们。例如：

```
<!-- 这是一个 HTML 注释 -->

<!-- 这
是另一个注释，
分成多行 -->
```

注释对开发人员或者要用到页面上代码的任何人都是有用的。通常情况下，随着时间的变迁网站的员工也是经常变换的，注释可以用来解释为什么要以这种方式来设置，或者提示这些代码是做什么的。

这在响应式网站中尤其重要，因为有太多的代码了。在代码中加入注释能让你更容易知道每部分代码都是做什么用的。当然，也不要走极端，添加注释与增加代码一样将使页面文件变大，这会增加下载时间。

你可能注意到，注释使用的语法类似于我们前面添加 HTML5 Shiv 到示例网站中的代码：

```
<!--[if lt IE 9]>
    <script src="files/html5shiv.js"></script>
<![endif]-->
```

以上代码实际上叫作条件注释（conditional comment），它们只能被 IE 浏览器识别（其他浏览器只将其当作普通的注释并忽略），可以用来给予 IE 特别的指示。

3.7 总结

所有的网页都是用 HTML 创建的，它是一种标记语言，使用元素（标记）为每一块内容赋予含义和上下文场景。

HTML 当前的最新版本是 HTML5。HTML 文档最开头的文档类型声明告诉浏览器你的网页是以哪个版本的 HTML 编写的。并非所有 HTML5 元素都已被支持，所以你需要知道哪些元素可以安全地使用，哪些元素需要额外的代码才能在老式浏览器中工作。

你需要遵循由 W3C 制定的 Web 标准，这样浏览器将按你期望的方式来显示你的网站。

你的 HTML 文档以文档类型声明开头，接着是 `<head>` 元素，其提供页面的相关信息，如页面标题（`<title>`），并告诉浏览器应该如何显示页面的信息，比如字符集（`charset`）和视口（`viewpoint`）。对于响应式网站，视口是确保页面能正确显示的关键。

紧随 <head> 元素之后的是 <body> 元素，其包含所有的页面内容。使用语义化 HTML 正确地表达每个页面元素的含义，这样不管是以何种设备或方式浏览内容，它都能正确地显示。

页面结构可以使用数个 HTML5 结构性元素来建立，如 `<header>`、`<footer>`、`<nav>`、`<section>` 和 `<aside>`。要使你的网站是完全可访问的，你还需要在这些元素上使用 WAI-ARIA 属性为老式的不能识别 HTML5 元素的屏幕阅读器提供导航帮助。

编写简洁的语义化 HTML 代码、分离内容与表现等，这些对于响应式网站尤其重要，因为有太多复杂的事要做。你还应该在你的代码中包含注释，这些文档化信息对于将来接手网站工作的人们是有帮助的。

在第 4 章中，我们将学习如何在网页中应用 CSS 样式，并为我们的示例网站添加一些样式。

第 4 章

响应式网站之CSS

HTML 赋予网站中所有内容以结构，CSS 则告诉浏览器以什么样式来显示这些内容。

如同第 3 章，对于已经熟悉 CSS 的人来说本章将是一个复习。不过，相比 HTML 我们会更加深入地介绍 CSS 的基本概念，因为它是构成响应式设计的代码基础。只有透彻理解层叠与盒模型等概念，才能真正明白如何创建一个响应式网站。

我们首先会介绍 CSS 的作用、CSS 的版本，以及如何通过浏览器前缀来确保新的样式属性可以在不同浏览器中被恰当地渲染，即使这些属性还处在测试阶段。

然后我们会看看如何在网站中添加样式。方法有许多，你可以在网页中直接嵌入样式，也可以使用样式表将样式应用于多个页面或整个网站，还可以对单个元素使用内联样式。

接下来，你将了解*层叠*（cascade）的概念，它是层叠样式表（CSS，Cascading Style Sheet）这个名称的由来。层叠决定了应用样式的顺序以及在样式规则发生冲突时浏览器如何进行选择。我们也会讨论在网站中如何使用层叠将样式规则之间的干扰降到最低。

之后，我们将重温*盒模型*（box model）的概念，其决定了元素如何显示在网页上：每个元素都被视作一个盒子，拥有宽度、高度、外边距、内边距和边框等属性值（有时是零）。我们也会讲到显示（display）和定位（positioning），它们将影响元素被放置于页面上的什么位置。

最后，我们将回到示例页面，添加一些基本的文字排版和布局样式，从而为页面赋予一些视觉定义。

4.1 CSS的工作方式

为了方便那些还不熟悉 CSS 的读者学习后续内容，本节将简要介绍样式表中的代码是如何组合的。虽然无法涵盖你能用 CSS 做出的所有组合变化，但也足够你了解 CSS 的作用了。

首先，每次你应用样式于一个 HTML 元素时，这称作一条 CSS 规则（rule）：

```
p { color: red; }
```

每条规则由两个独立的部分组成，*选择器*（selector）对应于要应用样式的 HTML 元素（只是没有尖括号）。在上面那条规则中，`p` 对应于 HTML 元素 `<p>`。该规则告诉了浏览器要如何显示页面上的段落。

在选择器之后，你可以有一个或多个声明（declaration）。它们包含在大括号中，定义了将应用到该元素的样式。

在声明中，*属性*（property）是你可以改变的元素特性，比如颜色或宽度。你需要设置一个*值*，比如橙色或 50%。每个属性后跟一个冒号，然后是属性的值，像前面例子中的声明 `color: red`，就是指定段落文本颜色为红色。

如果在同一条规则中有多个声明，它们之间用分号隔开：

```
p { color: red; font-size: 1.5em; }
```

最后一个声明（或者只有一个声明）后面的分号可以省略，但出于一致性的考虑，大多数开发人员还是会带上它。包含在大括号中的多个声明称为*声明组*（declaration group）。

通过使用类或 ID 选择器，样式也可以应用于元素的子集，：

```
.classname { color: blue; }
#idname { color: green; }
```

这两行代码将分别影响类属性为 `classname` 的元素（即 `<p class="classname">`）及 ID 属性为 `idname` 的元素（即 `<div id="idname">`）。

类和 ID 选择器的用途相似，允许你针对页面上的任何元素进行样式设定。但 ID 选择器只能应用样式于单个元素，而类选择器则允许你应用样式于任意多个元素。

你应该对类和 ID 使用描述性名称。虽然浏览器就像对 HTML 一样，并不会在意它们是否有意义，但当名称能够明显地表达它所对应的含义时，开发人员能更好地理解它。

示例：

```
<p class="intro">...</p>
<nav id="main">...</nav>
```

你也可以用后代选择器（descendant selector）来针对更具体的元素进行样式设定，这意味着某些在元素之中的元素：

```
.classname p { color: purple; }
```

在上面的代码中，我们告诉浏览器首先查找类属性为 classname 的元素，然后这些元素中的 <p> 元素的文本颜色应该是紫色（而对于那些类属性不为 classname 的元素中的 <p> 元素则无影响）。

如果你希望同样的样式也应用于多个选择器，可以使用逗号将它们组合在一起使用，而非创建两条独立的规则：

```
h1, h2 { color: green; }
```

这个例子告诉浏览器，<h1> 和 <h2> 元素的文本颜色都应该是绿色的。

当你编写 CSS 时，空格（后代选择器中的空格，比如前面例子中的 .classname p 除外）和换行都是可选的。

因此，一些看起来像这样的样式表，是完全正确的（尽管难以阅读）：

```
p{color:green}div{width:50%;backgroundcolor:blue}.classname{color:yellow}
```

通过删除空格（或优化样式表）可以减少 CSS 文件的尺寸（字节少一个算一个），这将使你的页面加载得更快（我们将在第 11 章中讨论如何使用软件自动删除空格）。

[小贴士]

优化过的样式表是很难阅读的，因为所有的内容都连在了一起。有一些在线工具可以在适当的位置添加空格和换行，从而方便阅读。你可试一下 Clean CSS（http://www.cleancss.com/），或者直接在网上搜索 CSS 优化工具（CSS optimizer）。

4.2 CSS版本

与 HTML 一样，CSS 也有不同的版本。CSS 第一版发布于 1996 年，仅比 HTML 第一版晚几年发布。最新的版本是 CSS3。

每一版 CSS 包含了所有可用 CSS 样式化的属性。属性是能够被样式影响的 HTML 元素的特性。这些特性包括元素的颜色、字体、大小及元素在页面上的位置，等等。

CSS3 引入了一些新的属性。CSS3 最显著的变化之一是增加了媒体查询，这使得响应式设计成为可能。你将在第 5 章更深入地了解媒体查询的工作方式。

还是和 HTML 一样，并不是所有的浏览器都能够在页面上渲染或显示最新版 CSS 中的所有属性，我会在书中标明哪些属性是你需要注意的。

另外还要注意，也并非所有的浏览器都会以完全相同的方式渲染 CSS 中的所有属性。你应该注意这种差异，并在不同的浏览器和设备上测试你的网站，以确保它能够如你所希望地那样显示出来。

图 4-1 中显示的是正常的 Mozilla 网站，而图 4-2 给出了该网站在去除了所有 CSS 后的显示效果。

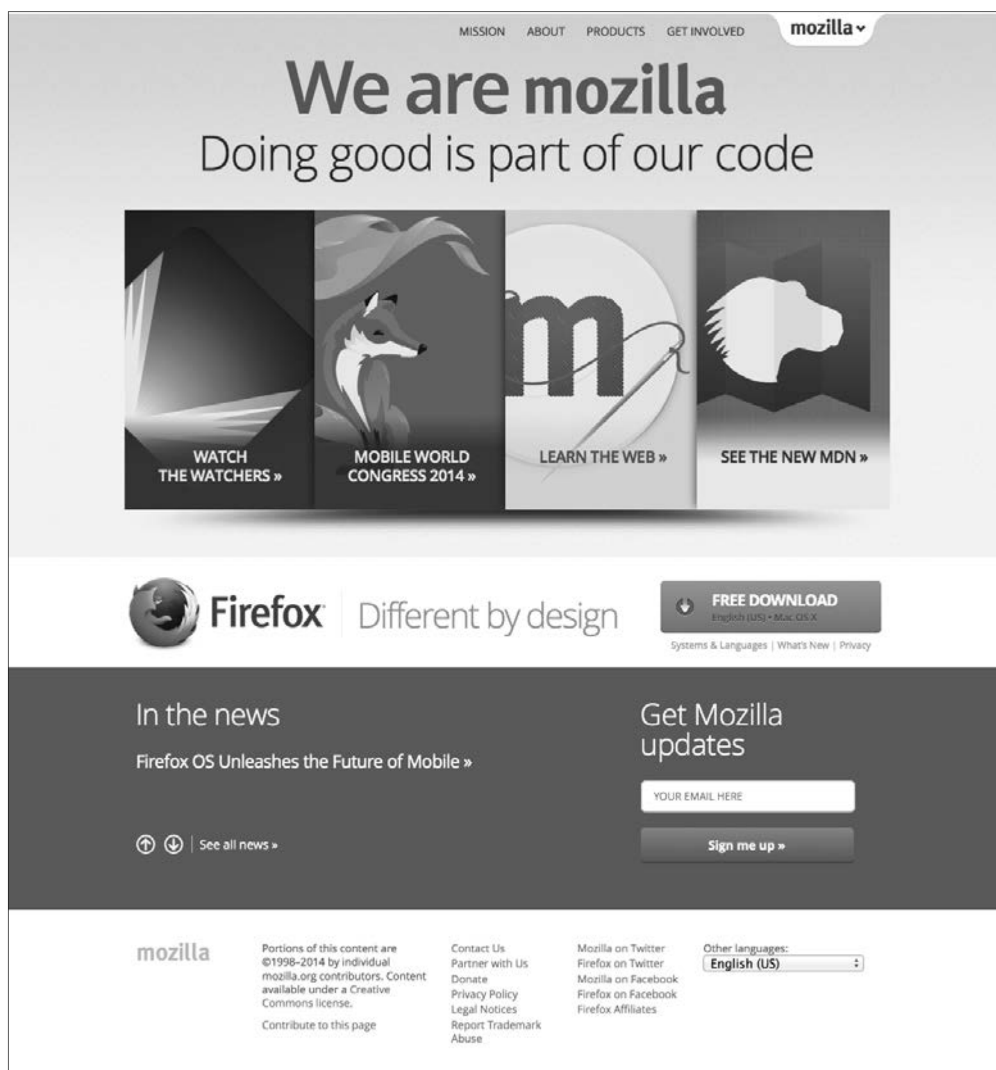


图 4-1：正常显示的 Mozilla 网站



图 4-2：去除了所有 CSS 后的 Mozilla 网站

浏览器前缀

CSS3 的部分规范仍在由 W3C 制定和被浏览器厂商测试。因此，对于任何新的样式，最终的规范可能与浏览器目前渲染它的方式会稍微有所不同。

为使开发者脱离进退两难的境地（不知道新的 CSS 属性代码在特定的浏览器中将如何显示），浏览器支持浏览器前缀（browser prefix，通常也称作*厂商前缀*，vendor prefix），使开发人员更容易控制如何使用 CSS 新属性。

我们实际上并不需要担心浏览器的差异性这一问题，因为它们大多数都是采用四大*浏览器渲染引擎*之一来显示网页的。要渲染一个页面，浏览器需要获取所有的内容（HTML 和图像）和样式（CSS）以及可能的 JavaScript，并把它们组合在一起来决定如何在屏幕上显示该页面。因此，所有使用相同渲染引擎的浏览器对一组特定的代码组合都会以相同的方式显示 。

下面介绍一下在前面提及的四大渲染引擎：

- WebKit 渲染引擎，为苹果公司的 Safari 浏览器和谷歌公司的 Chrome 浏览器所用；
- Mozilla 的渲染引擎 Gecko，为 Firefox 浏览器所用；
- 微软的渲染引擎 Trident，为 IE 浏览器所用；
- Opera 的渲染引擎 Presto，为 Opera 浏览器所用。

对于仍在制定中的样式，每个渲染引擎所采用的渲染方式可能是不同的，每个渲染引擎都会有自己的前缀化 CSS 属性对应于这些样式，它们是真正的属性的变体。

例如，连字符属性 `hyphens` 还未制定完成，所以它在四个浏览器渲染引擎中都有变体，而你需要在代码中列出每一种变体：

```
p {
    -webkit-hyphens: auto;
    -moz-hyphens: auto;
    -ms-hyphens: auto;
    -o-hyphens: auto;
    hyphens: auto;
}
```

这 5 条声明都包含一个不同的属性，最后的声明是无前缀的 `hyphens` 属性，这个属性的规范将由 W3C 来最终敲定。

前 4 条声明分别对应于列出的 4 种浏览器渲染引擎，你可以通过属性名的开头部分（`webkit`、`moz`、`ms` 或 `o`）来判断对应的渲染引擎。需要注意的是，所有的前缀属性都以连字符开头。

在这个例子中，如果 WebKit 渲染引擎还在测试 `hyphens` 属性，那么 WebKit 核心的浏览

器只能够识别 -webkit-hyphens 属性并渲染这条声明。对于不能识别的其他前缀属性或无前缀属性，它们则会将其忽略掉。

那么当规范被最终确定时又会发生什么呢？一旦 WebKit 渲染引擎实现了真正的 hyphens 属性，那么至少在短期内 -webkit-hyphens 属性仍会被以 WebKit 为核心的浏览器识别。不过，因为无前缀 hyphens 属性是定义在最后的，且它现在是有效的属性了，所以它会被 WebKit 浏览器识别，并覆盖掉定义在前面的 -webkit-hyphens 属性的设置。

大多数时候，在使用一个尚未最终确定的属性时，你需要包括（所有的）4 个前缀属性，如前例所示。

4 个浏览器前缀属性的顺序并不重要。许多人喜欢使用前面例子中的顺序（即从最长到最短），只是因为这样在样式表中看上去更加整齐，但无前缀属性应该总是放在最后。

你会发现，在这段代码中我把每条声明都单独放在一行中，而对其他地方的 CSS 我并没有这么做。这是一个例外，因为单独的行能够让你更容易确保包含了所有的前缀，并能一眼看到每个属性都包含相同的值（尽管在某些情况下，某些前缀属性可能需要使用不同的值）。

最终，你将不再使用每种属性的前缀版本，而只使用无前缀属性（不过，总是会有更多仍将需要前缀的新属性被引入）。

CSS3 中的几个新属性已经得到部分或全部主流浏览器的支持。在你的 CSS 中保留不再需要的前缀属性并不会妨碍页面的渲染，因为我们把非前缀属性放在了最后，它会覆盖掉前面的属性。当然了，在 CSS 文件中保留不再需要的前缀属性会增加一些不必要的字节。

如果一个 CSS3 属性在本书付印之际还是需要前缀的，我将会予以说明，不过在你读到这里的时候，这种状况可能已经发生了改变。

访问 Can I Use 网站（http://caniuse.com/）看看哪些 CSS 属性被哪些版本的浏览器支持，以及你是否需要使用前缀。

4.3 置CSS于何处

样式能够以数种不同的方式应用到你的网页，可以通过单独的样式表文件来应用，也可以直接包含在网页代码中。你的 CSS 代码放置的位置将影响到它如何应用于页面。

4.3.1 嵌入式样式

如果你想要样式仅用于网站上的某一个页面，可以通过 <style> 元素直接将它们添加到 HTML 文件的 <head> 元素中。只需将一条或多条 CSS 规则放置到 <style> 元素里，如

下所示：

```
<head>

<style>
p { color: blue; }
</style>

</head>
```

不过通常情况下，你不会让样式只应用于一个页面，而是希望将它们应用于整个网站。即使一个特定的样式只应用于一个特定的页面，你最好也把它单独放在一个样式表中。这样所有的样式表都在一个地方，很容易找到，特别是不止一个人在为网站编写代码时。

但是在给 HTML 格式的电子邮件编写代码时，你则应该使用嵌入式样式，因为邮件客户端通常不能导入外部样式表。

4.3.2 样式表

一般来说，你的 CSS 声明将被集中到一个或多个样式表文件中，你将从 HTML 文件中链接这些样式表。使用样式表意味着整个网站的字体、颜色等样式和布局都是一致的，并且一个样式只需声明一次，就能应用到网站的所有页面。

`<link>` 元素允许你链接文档（文件）到你的 HTML 页面，包括 CSS 和 JavaScript 文件。`<link>` 元素包含在 HTML 文件的 `<head>` 元素中：

```
<head>
<link rel="stylesheet" href="styles/mystyle.css">
</head>
```

任何一个网页都可以链接多个样式表。

此外，你也可以在 `<head>` 的 `<style>` 元素中添加一条 `@import` 规则来导入样式到页面中：

```
<head>

<style>
@import url(styles/mystyle.css)
</style>

</head>
```

就浏览器获取样式而言，链接和导入样式表实际上做的是同样的事情。但是，如何选择却可能对网站的性能产生影响，我们将在第 11 章讨论这个问题。一般来说，你应该使用 `<link>` 元素并尽可能地减少你的样式表。

4.3.3 内联样式

偶尔你可能想要直接在 HTML 中对一个仅出现一次的元素应用样式。

要做到这一点，只需为元素添加一个 style 属性（style="…"），在引号包裹的属性值中加入任何你想要的样式：

```
<p style="color: green; font-size: 2em;">This is a
paragraph.</p>
```

你应该避免在网站中使用内联样式，因为这会让你很容易忘记你的 CSS 都定义在哪里以及它们的作用，从而增加网站的维护难度。不过，在测试页面样式的修改效果时可以使用这种方式。

4.4 层叠

如果你已经能惬意地使用 CSS 了，那么你一定很熟悉本节的内容，但它们仍然值得再回顾一遍。

到目前为止，我们有了在外部样式表中的样式、<head> 元素中的嵌入式样式以及内联样式。我们还知道，样式可以被附加到元素、类和 ID 选择器中。那么，如果有多个相互冲突的样式分配给同一个的元素，浏览器应该选择哪一个呢？

CSS 对于以何种顺序来应用样式有非常详细的规则，这些规则就是层叠样式表（Cascading Style Sheet，CSS）中的*层叠*。层叠决定了哪些规则优先于其他规则。

层叠可能很难以理解，不过大多数时候你的 CSS 是直截了当的，会有什么效果也很清楚。我会以一个略微简化的版本来讲解层叠的作用方式，但是如果你要编写大量代码，那么应该做更加深入的学习。

4.4.1 层叠的作用方式

每个 HTML 元素都有若干应用于它的 CSS 属性。例如，<p> 元素的 CSS 属性包括文本颜色、背景颜色、字体大小、字体选择，以及是否粗体、斜体、有边框，是否有边框的样式，还有段落在页面的位置，等等（有十几种之多）。

当浏览器渲染网页时，它逐一查看每个元素并通过判别这些属性的值来决定如何渲染该元素。

浏览器按照一套非常具体的程序步骤（一种层级结构）来决定将哪些样式应用于特定的元素。这些步骤按从最高优先级到较低优先级的次序罗列如下：

(1) 标注为重要的规则；
(2) 内联样式规则；

(3) 包含 ID 选择器的规则；
(4) 包含类、属性和伪类选择器的规则；
(5) 包含元素和伪元素选择器的规则；
(6) 继承的规则；
(7) 默认值。

步骤的顺序通常称为 CSS *特殊性*（CSS specificity），它是从最特殊的规则（仅应用于一个元素的单个或几个实例）一直到最一般的规则（应用于一个元素的所有实例）。

对于页面上的每个元素，浏览器将按顺序遍历这个层级结构，直到为每个特定的属性找到一个样式为止。例如，我们告诉它需要知道一个段落的文本颜色（`color`）。它将首先寻找标注为重要的规则，然后是内联样式、包含 ID 选择器的规则，以此类推，直到它找出所有用于段落的颜色值，然后再根据层叠的规则来决定哪个属性值优先。

4.4.2 重要的规则

如果你希望一条特定的规则必须生效，可以使用 `!important` 来标注它。在你的常规样式或内联样式中都可以用它。在继续下一条规则之前，浏览器会首先在本条规则中查找是否存在 `!important` 标记。

在下面的例子中，`color` 属性被标注为 `!important`，而 `font-weight` 属性则没有。`!important` 标记放在分号之前，只对一条声明而不是整组声明有效：

```
p { font-weight: bold; color: blue !important; }
```

这样该段落将显示蓝色的文字，而不管是否在其他声明中对段落应用了别的颜色。

尽管 `!important` 看上去是一个有用的工具，但你应该仅在用类、ID 或其他选择器完全没办法来达成你的目的时才使用它。因为一旦应用了 `!important`，你的规则将是最高优先级的，也就不再有办法覆盖它了。

4.4.3 内联样式

如果浏览器没有找到任何标注为 `!important` 的规则来设置颜色，接下来就会搜寻内联样式。这些样式是直接附加于你 HTML 文档里的元素中的。内联样式优先于其他存在于外部样式表中或嵌入于文档 `<head>` 元素中的样式：

```
<p style="color: blue;">Paragraph text.</p>
```

4.4.4 ID选择器

如果没有内联样式，浏览器将寻找是否有直接的 ID 选择器（或者间接通过一个后代选择

器）应用于该元素：

```
<p id="example">Paragraph text.</p>

#example { color: blue; }
```

4.4.5　类、属性和伪类选择器

如果没有匹配的 ID 选择器，浏览器接着将寻找是否有类、属性和伪类选择器可以应用样式于该元素。同样，它可以包含在一个后代选择器中：

```
.example { color: blue; }
.example p { color: green; }
```

4.4.6　元素与伪元素选择器

如果没有类、属性或者伪类选择器应用样式于颜色，浏览器会接着寻找包含在元素或伪元素选择器中的样式：

```
<p>Paragraph text.</p>

p { color: blue; }
```

4.4.7　继承规则

到了这一步，如果浏览器仍未发现有样式可以直接应用颜色于我们的 <p> 元素，它就会寻找是否有继承的样式可用。

继承非常简单，也就是指如果一个元素没有样式应用于它，它将继承其父元素的样式：

```
<p>This is a paragraph with <strong>bold text</strong>.
</p>

p { color: blue; }
```

因为我们没有给 <strong> 元素分配一个颜色值，它将从其父元素，即 <p> 元素中继承颜色值。<p> 元素的颜色值被设置为蓝色，因此 <strong> 元素的颜色也将是蓝色。

不是每个属性都能被继承的，但什么属性可被继承通常是显而易见的。例如，字体大小是可被继承的，因为一旦你为 <p> 元素设置了字体大小，肯定是想要 <p> 元素中的所有内容都是相同的大小，即使是在 <strong> 或 <em> 元素中的内容。

但是，边框是不能被继承的。如果给 <div> 元素设置了边框，你并不会想要或希望 <div> 元素中的每个段落也都有自己的边框。

4.4.8 默认值

如果到现在还没有找到一个适用于 `<p>` 元素的颜色，不用担心，每一个元素都有默认值。

对于大多数属性，默认值是 CSS 规范的一部分。例如，任何文本元素的默认颜色都是黑色。如果你不指定颜色，浏览器将使用默认的黑色。

所以，如果真的没有任何样式应用文本颜色于我们的 `<p>` 元素，那文本颜色将是黑色。

4.4.9 发生冲突怎么办

我们还需要再多做一点解释。

在 CSS 规则的层级结构之外，浏览器还会特别查看一条规则包含的选择器数量。更多的数量等于更高的优先级（例如，一条具有两个 ID 选择器的规则优先于只有一个 ID 选择器的规则）。

如果两条规则仍然是平级的，浏览器不知道该选择哪一个，这个时候就会选择所有样式按顺序（从第一个样式表直至最后一个样式表）排列在一起时后出现的那条规则。

你在这里读到的是一个概要性的介绍，差不多涵盖了所有你需要了解的知识。但是如果你想对 CSS 特殊性这一概念有更多的了解，或者用它结合 CSS 来做更酷的事情，请参考 Vitaly Friedman 发表在 *Smashing Magazine* 上的文章“CSS Specificity: Things You Should Know”（http://coding.smashingmagazine.com/2007/07/27/css-specificity-things-you-should-know/）。

4.5 使用层叠

所有的这些规则可能看上去令人十分困惑，但实际上却很简单，因为是你来决定如何应用样式。

当你向页面添加样式时，你实际上是按我们刚才所讨论规则的相反的顺序添加。你总是希望样式能应运于尽可能广的范围，所以很少会遇到那些特殊的情况。

例如，你不可能为页面上的段落设置数十种文本颜色，以至于要去梳理众多相互冲突的样式规则。你可能只有一两条规则，并且知道如何准确地应用它们。下面是你在定义 CSS 时应该遵循的顺序：

(1) 默认值；
(2) 继承的规则；
(3) 包含元素与伪元素选择器的规则；

(4) 包含类、属性和伪类选择器的规则；

(5) 包含 ID 选择器的规则；

(6) 内联样式规则；

(7) 标注为 `!important` 的规则。

4.5.1 默认值和重置

首先讲解默认值。对于大多数 CSS 属性来说，默认值是显而易见的。以我们在第 3 章中所用到的 HTML 页面为例，在我们没有应用任何样式的情况下，文本颜色是黑色，标题文本大于段落文本，任何元素都没有边框。这些用的都是默认值。

不过，浏览器使用的默认值可能会与我们的预期有冲突，所以为了确保我们的初始环境相同而不管使用什么浏览器，我们将一些属性值重置为零。否则哪怕只有个别默认值不对，都可能影响到我们的设计，使得网站在不同的浏览器中有不同的显示效果。

重置

重置 CSS 需要我们将所有元素的内、外边距都设置为零，并设置某些字体属性为标准的基本字体大小。重置完后，我们再添加样式去设置我们想要设置的属性。重置所有属性的 CSS 代码通常称为 CSS *重置*或*样式表重置*。

举个例子，如果我们在“永远的大熊猫”网站使用 CSS 重置，得到的显示效果如图 4-3 所示。

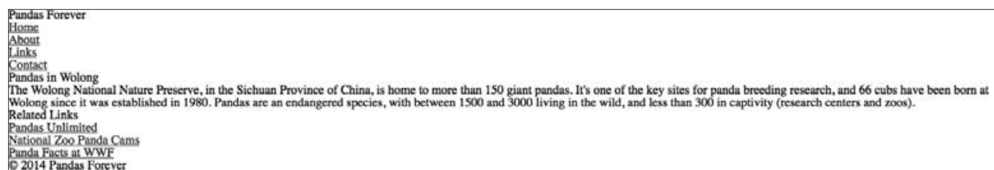


图 4-3：我们的示例页面，所有的样式被重置

还是以前的好看，对吧？不过别担心，我们将一点点修正它。

找出你的 CSS 中所使用的所有元素，然后编写样式声明来重置它们，这可不是一件轻松的事。幸运的是，已经有人为我们做好了这件事，他们创建的 CSS 代码块包含了你需要重置的所有样式，你可以简单地复制并粘贴到样式表中。

一个最受欢迎的 CSS 重置方案是 Eric Meyer 的“Reset CSS”（http://meyerweb.com/eric/tools/css/reset/）。它不受版权限制，因此你可以随意地使用与修改它。

另一个可选方案是出自 Nicolas Gallagher 的“Normalize.css”(http://necolas.github.com/normalize.css/) 。

你还可以创建自己的重置方案，详情请见 Jeffrey Way 在 Nettuts+ 上撰写的“Quick Tip: Create Your Own Simple Reset.css File”（http://net.tutsplus.com/tutorials/html-css-techniques/weekend-quick-tip-create-your-own-resetcss-file/）。

为确保 CSS 重置不会覆盖你的其他样式，它必须最先被浏览器渲染。所以它应该是你的 <head> 元素中链接的第一个 CSS 文件，如果你没将它单独放在一个文件中，那就应该放在你的第一个 CSS 文件的开头处。

4.5.2 继承规则

下一步我们要看的是继承规则。你的网站上的大多数样式都会被继承。

因此，如果你设置 <body> 元素的文本颜色是粉红色，那么网站上的所有文本都将是粉红色的。你无需费心去设置段落、列表、<div> 等元素的文本颜色，除非你希望它们显示粉红色之外的颜色：

```
body { color: pink; }
```

但有一个例外，链接元素（<a>）不会继承颜色设置，默认情况下它们是蓝色的（访问过的链接是紫色的），这样它们在页面上有不同的视觉效果。你可以保留它们为蓝色 / 紫色，或者通过 <a> 元素给它们分配不同的颜色。

另外，不要添加多余的样式声明。如果设置 <body> 元素的文本颜色为粉红色，那就不用再添加一条规则使所有 <p> 元素的文本颜色也是粉红色。因为它们会继承粉红色的设置，你不需要再添加额外的代码。

4.5.3 元素规则

应用样式首先要从最大的范围开始。例如，如果你想要所有文本都使用 Helvetica 字体，那么应该对 <body> 元素应用这个样式规则，这样页面上的所有其他元素都会继承这个设置。

如果你想要所有的一级标题使用 Georgia 字体，那就应用该样式于 <h1> 元素，这样你的所有一级标题将使用 Georgia 字体，其他元素则仍是 Helvetica 字体。

不管你要应用什么样式，试着找出它可以应用的最高层级的元素。

在我们的网站中，我们要改变标题的文本大小，使它比页面上的其他文本要大，然后加粗：

```
h1 { font-size: 2em; }
h2 { font-size: 1.3em; }
h1, h2 { font-weight: bold; }
p, ul { font-size: 1em; }
```

em 是一个相对单位，2 em 意味着 `<h1>` 元素的文本将是基本字体大小（网站上所有其他文本的字体大小）的两倍。我们将在 4.8.1 节更详细地讨论它。

虽然段落和列表会是默认的字体大小，但我们还是将它们包含在 CSS 中以免发生冲突，因此我们给它们设置的值是 1 em。

4.5.4 其他规则

如果某一元素的样式需要特别设置，有别于该元素的其他实例，你可以使用以下几种方式。例如，如果你想要 `<nav>` 元素中的链接字体变大，可以这样：

```
nav li { font-size: 1.5em; }
```

网站上所有其他的 `<li>` 元素仍是继承的字体大小（除了 `<nav>` 元素中的 `<li>` 元素），所以你会看到在图 4-4 中，"Related Links"（相关链接）部分中链接的字体大小并没有改变。

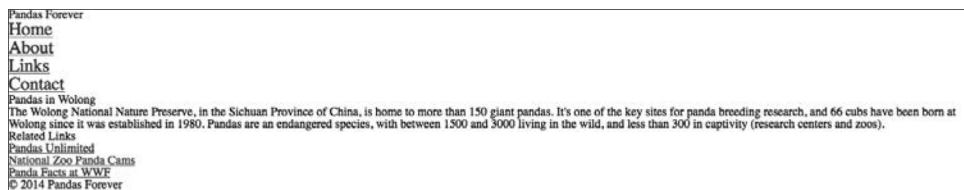


图 4-4：导航中的列表项字体尺寸变大，但其他列表项字体尺寸保持不变

我们应像前面的例子那样优先使用后代选择器，然后再考虑为元素添加类属性并通过类选择器将它们与其他元素区分开。

但是除非你真的需要它们，否则不要使用后代选择器和类选择器。在上面的例子中，因为链接实际上是 `<nav>` 中唯一的文本，我们可以直接应用样式于 `<nav>` 元素而无需使用后代选择器。下面的代码具有同样的效果：

```
nav { font-size: 1.5em; }
```

层叠规则中最特殊的是 ID 选择器、内联样式和 `!important`，你应该避免使用它们，除非你真的知道自己在做什么。

4.5.5 保持简单

你可能想知道为什么需要保持 CSS 尽可能得简单。为什么不加入类选择器和 `!importants`，即使他们能实现目标?

原因是，对于任何网站，你几乎肯定会在后来修改 CSS，不管是细微地调整还是彻底地重写。最初的 CSS 越简单，就越容易进行修改，尤其是进行修改的人并非原开发者时。

而且正如我之前提到的，本书只是告诉你非常基本的 CSS 工作方式，实际编程往往要复杂得多。

如果网页上显示的某些内容与预期有出入，可能问题出在层叠渲染的样式上。请测试并修改这些样式，直到你弄明白发生了什么。学习事物工作方式的最简单办法就是尝试不同的变化然后看看发生了什么。

你需要在不同的浏览器和设备上测试你的网站，以保证所有浏览器和设备应用的 CSS 有着相同效果。我们将在第 8 章中讨论如何测试。

4.6　注释

与 HTML 一样，你也可以对 CSS 文件添加注释，不过它们所使用的注释符号不同。

浏览器会忽略掉注释中的一切内容。

在样式表中使用注释，通常的方式是提供一个描述性或解释性的内容，比如什么是干什么用的，或对修改内容做个记录：

```
/* 这是一个 CSS 注释 */

/* 它们可以是一行或多行，
也可以与你的样式代码在同一行 */ h1 { color: #7b0000; }
```

4.7　组织你的样式表

除非样式需要以特定的顺序来应用，否则样式表中的样式几乎可以是任意顺序的。但最好还是以系统的方式组织它们，这样在以后也能更容易地检索。一旦你开始添加媒体查询进来，就需要知道所有的样式都在什么地方。

我们首先要重置 CSS。这可以是单个重置文件，也可以是你主样式表中最开始的几行代码。

有些人喜欢按页面或站点区块来编排自己的样式表，并用注释标注出样式表的每一部分。你可以用破折号或其他符号在注释中画“线”以区分样式表中的各个部分，这样更容易找到特定的代码：

```
/* 主要内容 ----------------------------- */

styles for typography, layout, etc. in main content

/* 页眉 ----------------------------- */

styles for typography, layout, etc. in header

/* 页脚 ----------------------------- */

styles for typography, layout, etc. in footer
```

其他人喜欢把类似的样式放在一起，不管它们所处的位置或区块，比如：

```
/* 文字排版 ----------------------------- */

p { ... }
header p { ... }
footer p { ... }
.classname p { ... }

li { ... }
footer li { ... }
.classname li { ... }

/* 布局 ----------------------------- */

header { ... }
footer { .... }
```

你可以通过浏览器查看任何一个网站的 HTML 和 CSS 文件，看看在这些网站的 CSS 文件中，样式是如何组织安排的。并不存在唯一正确的组织方法，但在网站中你应该保持组织方法的一致性。

4.8 盒模型

CSS 布局中最重要的概念是“盒模型”。也就是说，页面上的每一个 HTML 元素都是一个矩形盒子，每个盒子都可能有（或没有）边框、内边距（盒子边界内的空白）和外边距（盒子边界外的空白）。

如图 4-5 所示，位于元素盒子正中的是内容，它可以是文字、图像或子元素（比如 `<div>` 元素中的段落元素）。元素被内边距环绕，接着是边框，最后是外边距，它们的宽度和高度都可以设置成零（宽度和高度值都为零的边框将是不可见的）。

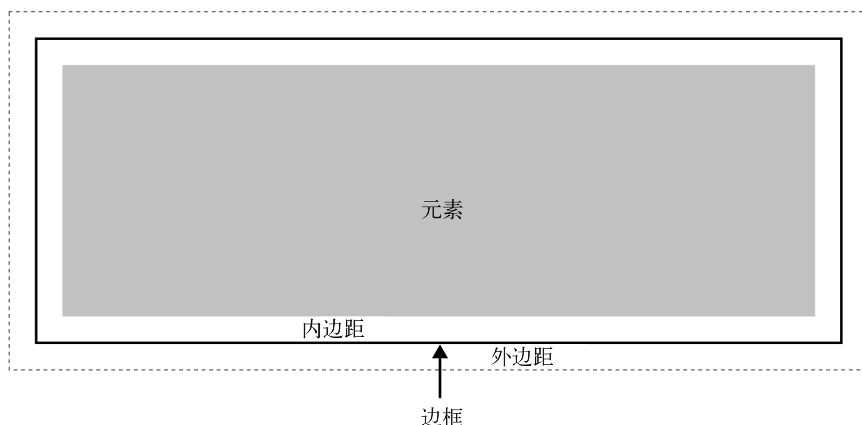


图 4-5：元素四周被内边距、边框和外边距包围

4.8.1 度量单位

元素的高度、宽度、内边距、外边距和边框可以使用任何一种度量单位进行设置。

习惯上，我们使用绝对度量单位，以保证一个元素在屏幕上的精确位置。在响应式设计中，我们通常使用相对度量单位，这样每个元素都可以对视口的大小做出响应。

下面是最常见的相对度量单位。

- `%`
  表示包含元素的百分比。
- `em`
  表示元素的字体大小。
- `rem`
  表示文档的字体大小（即 `html` 元素）。

下面这些是最常见的绝对度量单位。

- `px`
  像素是一个绝对度量单位，但在不同设备上它们不一定是一致的。
- `in`、`cm` 和 `mm`
  映射到像素的物理度量单位，但在不同设备上也不一定相同。一英寸是 96 像素，一厘米是 37.8 像素，一毫米是 3.78 像素。
- `pt`
  一磅等于一英寸的 1/72，它在印刷设计中更为常见。在网站中它们可以用于生成一个打印样式表。

4.8.2 em

在响应式设计中你会大量将 em 用作度量单位。

在传统的金属印刷排版中，em 是指包含一个浮雕字母的金属板的尺寸，它的宽度必须能容下最宽的字母——大写的 M。

许多人认为在数字世界中 em 也是基于字母 M 的宽度，其实不是。在数字世界中，em 就等于元素的字体大小，不会随所选择的字体而改变，如图 4-6 所示。

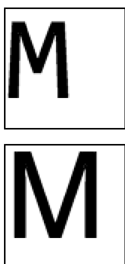


图 4-6：高度和宽度正好都是 1 em 的盒子

4.8.3　高度与宽度

在盒模型中，每个元素都有一个高度和宽度属性，某些时候我们可以用 CSS 改变它们。

行内元素有一个继承的高度和宽度，不能被覆盖。在默认情况下，一个行内元素的宽度和高度只是恰好能容下其所要显示的内容，如图 4-7 所示。

A link is an inline element and only as
wide and tall as it needs to be.

图 4-7：行内元素宽度和高度只是恰好能容下其所要显示的内容

块级元素默认情况下具有包含它的元素的 100% 的宽度，即使内容没有填满空间。

而默认情况下的高度是其包含内容所需要的高度。例如，一个段落元素的高度必须能够包含它所有的文本，即使自动换行成多行文本，如图 4-8 所示。

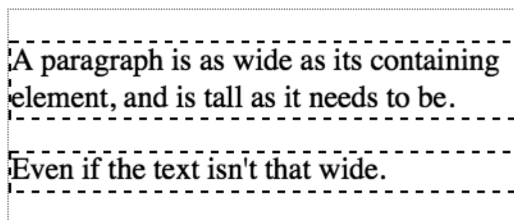


图 4-8：块级元素默认情况下与它们的包含元素一样宽，高度则是包含的内容所需要的高度

你可以用 CSS 改变块级元素的宽度或高度：

```
div { width: 75%; height: 200px; }
```

记住，一个百分比宽度是基于包含元素的宽度，所以，在上面的代码中，`<div>` 的宽度将是其包含元素宽度的 75%。

4.8.4 外边距与内边距

每个元素都有外边距，规定了页面布局中元素外围的空白区域。每个元素也有内边距，规定了元素内部环绕内容的空白区域。内、外边距所产生的效果通常是类似的，除非元素有边框（边框位于内、外边距之间）。

浏览器对于每个元素都有一个默认的内、外边距，但浏览器与浏览器之间，这个默认值并不总是一致。这就是早先时候我们为什么要使用 CSS 重置来设置每个元素的内、外边距为零。这样，你就不用担心在不同的浏览器中有不同内、外边距值，使得布局与你的预期相符。

一般来说，在响应式设计中，你将使用百分比设置元素的水平（左 / 右）内边距和外边距，这样布局才可以对视口的大小做出响应。在大屏幕上，你有足够的空间在内容四周设置内外边距。但是在小屏幕上，你肯定不想因为巨大的边距而浪费宝贵的屏幕空间。

垂直（上 / 下）内边距和外边距可以用像素来设置，因为它们不需要响应视口的大小。

元素的 4 个边都可以有不同的内、外边距值。

下面的例子分别设置元素四边的边距。按顺时针方向，分别是上、右、下、左四个方向（记住这个顺序的简便方法是用助记符号 TRBL 或 trouble 来记）：

```
div {
    padding: 1px 2px 3px 4px;
    margin: 1px 2px 3px 4px;
}
```

在这个例子中，上边的内、外边距是 1 个像素，右边是 2 个像素，下边是 3 个像素，左边是 4 个像素。

如果相对两边的边距或所有四边的边距的值是相同的，可以合并它们。

你仍然可以使用同一助记符号 TRBL，缺少的值将从相对的那边复制。所以在下面的例子中，上边距为 1 个像素，右边距为 2 个像素，下边距复制上边距的值，一样是 1 个像素，左边距和右边距也一样（2 像素）：

```
div {
    padding: 1px 2px;
    margin: 1px 2px;
}
```

如果有三个数值，仍遵循这个逻辑。如果你给出了上、右、下边距的值，而缺少的左边距的值将复制它相对的右边距的值（2 像素）：

```
div {
    padding: 1px 2px 3px;
    margin: 1px 2px 3px;
}
```

如果只给出了一个数值，它将用于盒子四个边的所有边距：

```
div {
    padding: 1px;
    margin: 1px;
}
```

你也可以单独地设置一边，例如，如果你已经设置了所有段落的外边距，但对一个特殊的段落想要覆盖掉上边的外边距值：

```
p { margin: 5px 10%; }
.other p { margin-top: 10px; }
```

你可以用 `padding-top`、`padding-right`、`padding-bottom`、`padding-left`、`margin-top`、`margin-right`、`margin-bottom` 和 `margin-left` 单独设置任意一边。当然，如果这些属性被应用到同一个元素，你应该将它们合成为一个声明，如前所述。

如果你想要一个块级元素在其包含元素内居中，你可以通过设置元素的左、右外边距值为 auto 来实现这个效果。这将使元素保持在其容器元素的中间位置，留出的空间将由左、右外边距平分，从而有效地让元素居中，如图 4-9 所示。

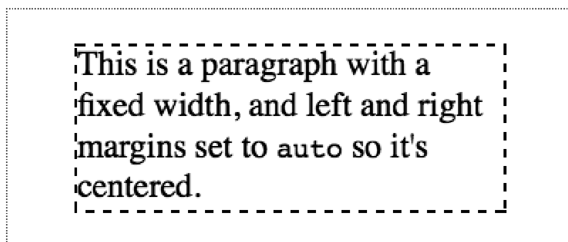


图 4-9：设置一个块级元素的左、右外边距为 **auto** 将使其在包含它的元素内居中

这个视觉效果仅在你要居中的元素的宽度小于 100% 时可见，否则，它只是填满整个空间罢了。

4.8.5　边框

边框位于内、外边距之间，为元素提供一个轮廓。

在声明一个边框时，你需要指定边框线的宽度、样式和颜色。

边框宽度通常以像素为单位声明。

边框样式值（`border-style`）包括 `solid`、`dotted`、`dashed`、`double`（双线），`groove`、`ridge`、`inset` 和 `outset`（后 4 个代表不同的方向 3D 边框）。你可以参考图 4-10 的示例。

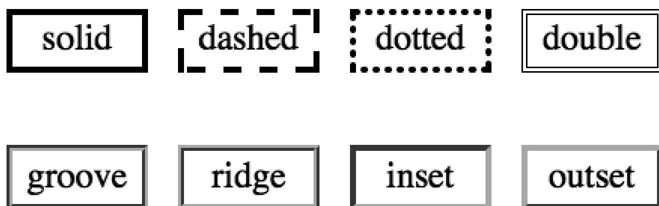


图 4-10：边框样式选项

你可以以任意顺序在一个声明中定义所有的边框值。下面的代码设置

元素的边框宽度为 1 像素，样式为实线，颜色为红色：

```
div { border: 1px solid red; }
```

你也可以对这三个属性进行单独地设置：

```
div { border-width: 2px; }
div { border-style: dotted; }
div { border-color: green; }
```

你也可以单独地设置元素的每一条边：

```
div { border-left-width: 2em; }
div { border-right-style: inset; }
div { border-top-color: blue; }
```

诸如此类，上、下、左、右四条边的这三个属性都可以设置。

4.8.6 盒子大小

现在你已经学会了如何设置盒子的各种属性，不过，你需要知道，在网页上计算盒子的大小是件有点复杂的事。内边距、边框和外边距是否被作为元素宽度、高度的一部分计算在内，这要视情况而定。

实际上有两个不同的方法来告诉浏览器如何计算。CSS3 中的新属性 `box-sizing`，允许你在 `border-box`（边框盒）或者是 `content-box`（内容盒）这两种方法之间选择其一。

对于 `border-box`，一旦你设置了你的块级元素的宽度，你对内边距和边框所设置的宽度值都将计算在这个宽度内。相反，使用默认的 `content-box`，内边距和边框的宽度值都不计算在盒子的宽度中。

`border-box` 通常更容易使用，但它却不是默认值。所以要使用它，你需要设置元素为 `box-sizing: border-box`。

这个属性并不完全被所有浏览器支持，所以你必须使用厂商前缀。IE8+ 和 Opera 已经支持

这个属性，所以你只需要针对 WebKit 和 Mozilla 使用前缀属性：

```
div {
    -webkit-box-sizing: border-box;
    -moz-box-sizing: border-box;
    box-sizing: border-box;
}
```

你可能想要将 border-box 应用于你的整个网站。实现此的代码是：

```
*, *:before, *:after {
    -moz-box-sizing: border-box;
    -webkit-box-sizing: border-box;
    box-sizing: border-box;
}
```

星号表示 CSS 将应用于所有的元素。

[小贴士]

box-sizing 在 IE7 及更早的版本中是不起作用的，所以即使你设置了 border-box，它仍将以默认的 content-box 方式来渲染。在测试网站时，你可能会需要添加额外的 CSS 来修复所有的布局问题。你也可以在 HTML5 Please（http://html5please.com/#box-sizing）上找到一个腻子脚本（polyfill）来使 box-sizing 在 IE6 和 IE7 上工作，其作者还发布过 HTML5 Boilerplate、Modernizr 和 CSS3 Please 等工具作品。

在图 4-11 中，两个段落的宽度都是 300 像素，每边的内边距都是 20 像素，边框宽度是 5 个像素。在第一个段落中，内边距和边框是计算在 300 像素宽度中的。在第二个段落中，内边距和边框则不计算在 300 像素宽度之中。

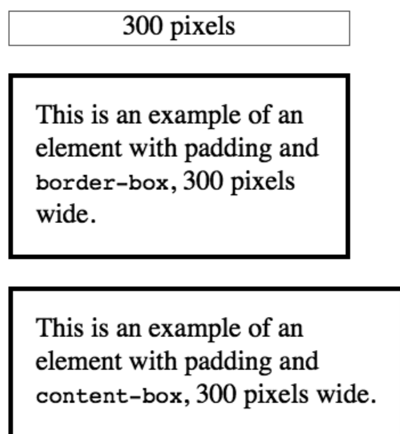


图 4-11：一个块级元素呈现出的不同的显示效果，只是因为在样式中设置 **box-sizing** 属性的值是 **border-box** 或 **content-box**

4.9 显示

HTML 元素的一个关键属性是 display，它会影响一个元素占据屏幕空间的方式。大多数元素的这个属性的值都是 block 或 inline。区别正如它们的名字，块级元素呈现为堆叠的块，而行内元素（也译为内联元素）呈现为文本流，如图 4-12 所示。

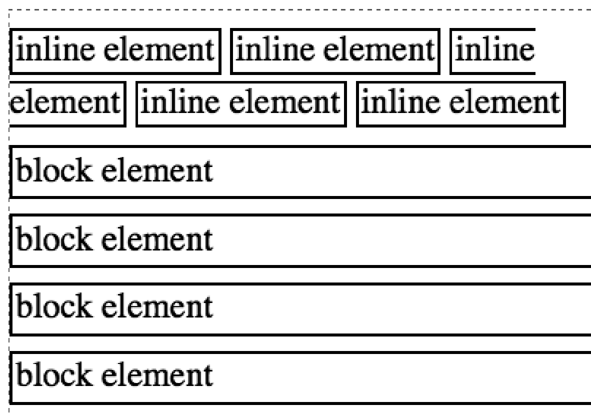


图 4-12：行内元素以文本流显示，而块级元素是堆叠的块

Display 是一个 CSS 属性，你可以对任何元素的这个属性值进行修改，不过你应该先了解你正在使用元素的默认 Display 值是什么。

块级元素的一个例子是段落。请看图 4-12，把每一个块级元素想象成是一个段落。每一个段落自动新起一行，而它后面的元素不管是什么也同样会新起一行。其他块级元素包括诸如 <nav> 和 <article> 这样的结构化元素、分级标题和 < div> 元素。

除非你手工设置成不同的宽度，否则块级元素总是水平填充包含它们的元素。因此，一个存在于 <body> 元素中的块级元素是与浏览器窗口一样宽的。每个块级元素自己新起一行，也会使后面跟着的元素新起一行，不管是不是块级元素。

而行内元素则是插入在其他元素的文本中间的。例如，在段落文本的中间，可能有一个行内元素 <strong>，仅包含几个设置成用粗体显示的单词。其他的行内元素包括 <em>、<span> 和链接。

行内元素的大小由其包含的内容决定。行内元素通常包含在段落或其他块级元素中，它自身不会新起一行，也不会使后面的文本新起一行。

除了块级元素和行内元素，还有列表项（list-item）元素，其对应于项目符号列表中的 ，其表现在大多数方面类似于块级元素。此外，表格、表格行和表格单元也都有它们自己的 display 属性。

`display` 属性还有一些其他的不大会用到的值，你可以在 Mozilla 开发者网络上的 display 分栏下查阅到完整的列表（https://developer.mozilla.org/en-US/docs/CSS/display）。

虽然对于大多数元素你会保持默认的 `display` 值，但仍可以用 CSS 改变它。例如，常见的将列表项的 `display` 属性从 `list-item` 更改为 `inline`，这样它们将排列成水平的一行，而不是垂直的一列：

```
li { display: inline; }
```

4.10　定位

定位是你能将元素放置于页面上指定位置的关键。如果你不是一个程序员，这可能稍微有点难以理解，不过不用担心，如果你不能完全明白它的工作原理，只需理解这些定位方式之间主要的差别就行。

默认情况下，页面内容的流向，对于行内元素是从左向右，并在必要时自动换行，而对于块级元素则是自上而下，就像图 4-12 所显示的那样。但是我们可通过设定位置（`position`）属性为不同值来改变这种默认状况。

4.10.1　静态定位

默认的定位方式是静态（`static`）定位。元素如你所期望的那样，出现在它们应该出现的地方，如图 4-13 所示。

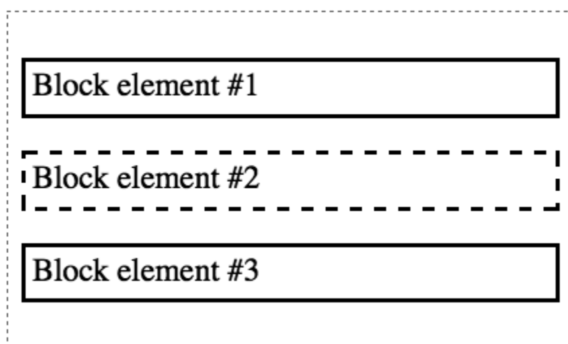


图 4-13：具有默认定位方式的块级元素像往常一样以堆叠的方式显示

4.10.2　相对定位

元素可以被赋予一个相对（`relation`）位置，它会忽略页面上的其他内容流，移动到一个不同的位置。例如，在图 4-14 中，第二个块采用了相对定位，相对于它原来的位置向下

移动了 40 像素，向右移动了 40 像素：

```
p.2nd { position: relative; left: 40px; top: 40px; }
```

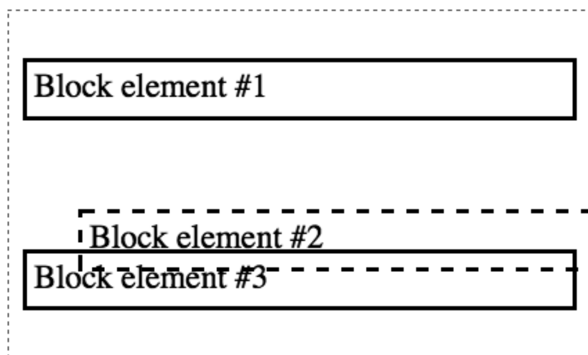


图 4-14：第二个块采用了相对定位

浏览器会保留元素原来所占用的空间，然后按指定的方向相对地移动元素。

正如你所看到的，它只是与页面上的其他元素相叠加，而不是将其他元素挤走来获得空间。在屏幕的右边，2 号块的边缘已经没有空间再向右移动 40 像素了，所以它的一部分移出了屏幕，用户看不到了。

注意，相对定位后的元素是叠加在页面的其他元素上方的。

要对一个元素相对定位，你需要设置 `position` 属性值为 `relative`，然后还要定义位置的变化，是水平移动，还是垂直移动，或两个方向同时移动。

4.10.3 绝对定位

绝对定位不同于相对定位。浏览器会按你的请求移动元素，但它不会再为元素保留原来的空间。

不同于相对定位元素是相对于它自身原来在内容流中的位置来进行定位的，对于绝对定位元素，浏览器是使用包含该元素的第一个非静态定位的元素来对其进行定位的。

在下面的例子中，三个块都是包含在一个 `<div>` 元素中的：

```
p.2nd { position: absolute; left: 40px; top: 40px; }
```

2 号块采用了绝对定位，所以它被其他元素流忽略。取而代之，它被放置到从包含它的 `<div>` 元素的左上角向下移动 40 像素，向右移动 40 像素的地方。

因为没有保留空间给它，它只能叠加在其他元素之上，如图 4-15 所示。

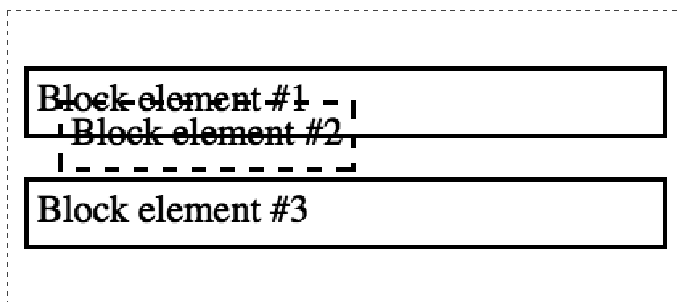


图 4-15：第二个块采用绝对定位

你会对到图 4-15 中的绝对定位元素感到奇怪，它的宽度只是与其内容一样宽，而不是像一个正常的块级元素那样，是整个包含元素的宽度。是的，绝对定位的块级元素不再像其他块级元素那样有默认的 100% 宽度。

如果有更多的文本在采用绝对定位的元素中，它们将一直扩展到它的包含元素的右边界，然后文本会自动换行。

[小贴士]

你可能已经注意到这些案例中有些元素是重叠的。对于重叠的元素，要改变它们的叠放次序，你可以为元素指定 z-index 属性值，具有最高值的元素将在最前面，你也可以使用负数：

```
p.1st { z-index: 50; }
p.2nd { position: relative; left: 40px; top: 40px;
z-index: 25; }
```

4.10.4　固定定位

固定定位与绝对定位非常相似，但元素是相对于整个页面来定位的，而不是相对于它的包含元素。所以，在前面的例子中，如果我们使用固定定位，2 号方块将会相对于网页的左上角向下移动 40 像素，向右移动 40 像素。

任何具有固定位置的元素将一直停留在某个位置，哪怕是用户滚动页面，所以如果你想要一个导航栏或一其他元素能总是停留在页面的顶部，使用固定定位就能实现。记住，其他元素将忽略固定定位元素，因此它们之间很可能会重叠。

固定定位在移动浏览器中的行为有点古怪。Brad Frost 在文章“Fixed Positioning in Mobile Browsers”(http://bradfrostweb.com/blog/mobile/fixed-position/）对此对此进行了详细说明。

4.10.5 定位元素的度量单位

当设定一个元素的相对、绝对或固定位置时，是相对于上、下、左、右四个方向来给定偏移值。

你可以使用 `left` 或 `right` 来（但不能同时指定两个）水平移动，`top` 或 `bottom` 来（也不能同时指定两个）垂直移动。如果你不小心给了它一组相对的方向（比如，同时给出 `left` 和 `right`），浏览器会忽略第一个声明。

你有时并不需要同时在水平和垂直方向上移动一个元素，这时你只声明其中之一的属性值即可。

注意，当设定方向时，设定值表示的是起始方向，而不是目标方向，所以 `left:40 px;` 是表示将方块从其原始位置的左手边向对边移动 40 像素，实际上也即是向右移动。

你也可以使用负数。但是负数在相对定位和绝对定位（及固定定位）中所产生的影响是不一样。如果你采用相对定位，一个元素向左移动 40 像素与向右移动 −40 像素是同样的效果。向上和向下移动也是如此，因为你只是在相对于元素自身在移动。

而绝对定位和固定定位就不一样了，因为它们不是相对自身移动。如下面的代码所示，绝对定位方式中，让元素从左上角向右且向下偏移 40 像素，那么在水平方向上它将位于包含元素左边线向右 40 像素的位置（参见图 4-16）：

```
p.2nd { position: absolute; left: 40px; top: 40px; }
```

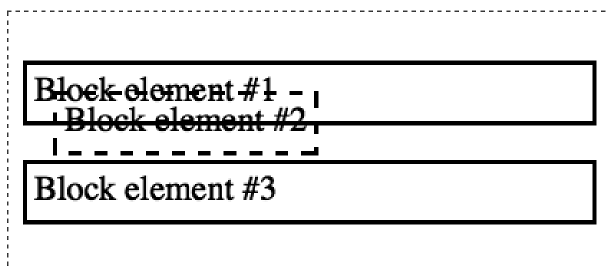


图 4-16：第二个长方块采用绝对定位，从左上角向右且向下偏移 40 像素

再来看负数的情况，对 2 号方块向右边移动 −40 像素，是要先将它与包含元素的右边线平齐，然后再向右移动 40 个像素（见图 4-17）：

```
p.2nd { position: absolute; right: -40px; top: 40px; }
```

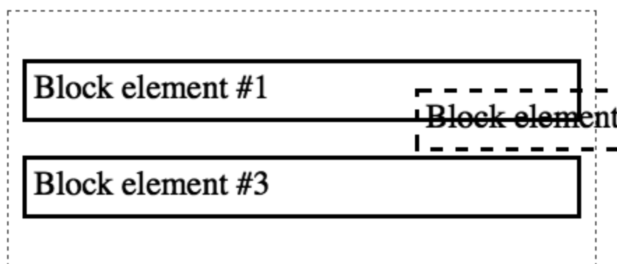


图 4-17：第二个方块从右边界向左移动 –40 像素后的位置

定位可能难于理解，最好的学习方式是建立一个测试页面，并对所有的这些属性进行尝试，这样能更好地了解它们是如何工作的。

4.11　浮动和清除

浮动（`float`）和清除（`clear`）属性也允许你改变一个元素在页面上的位置，尽管它们是独立于 `position` 属性的，通过浮动，你可以使内容流环绕着一个特定的元素。

例如，假设你有一个图片在两个段落之间，如以下代码所示（参见图 4-18）：

```
<p>...</p>
<img src="image.jpg">
<p>...</p>
```

The Wolong National Nature Preserve, in the Sichuan Province of China, is home to more than 150 giant pandas. It's one of the key sites for panda breeding research, and 66 cubs have been born at Wolong since it was established in 1980.



The Wolong Nature Reserve covers around 200,000 hectares (772 square miles) in the Ming Shan mountain range, in the ABA Tibetan Autonomous Region, near the Tibetan Plateau. Elevations in the reserve range from 1555 to 4600 meters (4,000 to 11,000 ft).

图 4-18：图片在两个段落之间未浮动

你不希望段落间出现空白，你想要让文本填满空两段之间的空白区域，如图 4-19 所示。



图 4-19：图片向左浮动

使用 `float` 属性告诉浏览器将元素浮动到它的包含元素的左边或右边，后继的元素，不管是块级元素还是行内元素，都将环绕浮动的元素。

当然，你并不想使网站上的所有图片都是浮动的，所以你需要以某种方式来区分图片，比如使用 `class` 属性。你可以定义一个 `floatleft` 类选择器，将其应用于你想要向左浮动的元素：

```
.floatleft { float: left; }
```

记住，所有后继元素均将环绕浮动的元素，直到没有更多的空间。最终有可能出现你并不想要的环绕效果，比如图 4-20 中的标题。

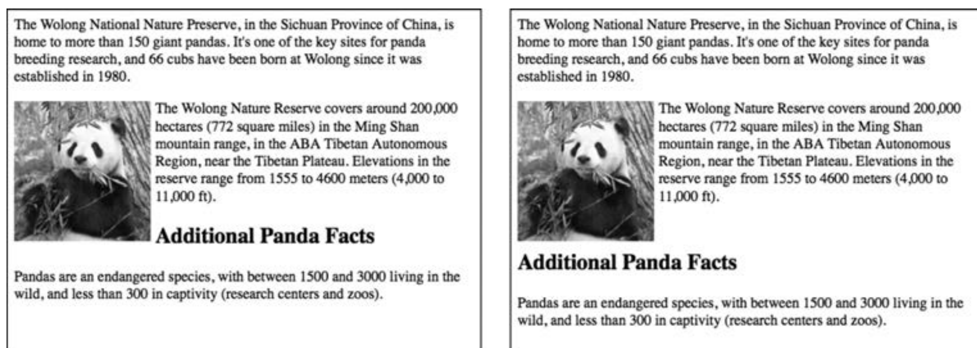


图 4-20：在右边的例子中，标题被清除了浮动状态，并新起一行至浮动图片的下方

如果你想取消浮动，仅需将 `clear` 属性应用于你要取消浮动的第一个元素（元素将新起一行）之上：

```
ul { clear: left; }
```

当你应用 `clear` 属性时，`left` 或者 `right` 属性值需要与前面的 `float` 属性值相匹配。

4.12 基本样式

回到“永远的大熊猫”这个网站，让我们添加一些基本的样式使它看起来更漂亮。

首先，用简单而不复杂的方式来做这件事。我们将用到本章之前讲到的 box-sizing 属性，我们将它应用于所有的元素：

```
*, *:before, *:after {
    -moz-box-sizing: border-box;
    -webkit-box-sizing: border-box;
    box-sizing: border-box;
}
```

我们设置了分级标题、段落的字体大小以及列表的样式，并使分级标题字体加粗。接下来，我们将添加一些内、外边距以使各元素间稍微分开些，并在列表项上加上圆点项目符号，如你在图 4-21 中所见：

```
h1 { font-size: 2em; }
h2 { font-size: 1.5em; }
h1, h2 { font-weight: bold; margin: 5px 0; }
p { font-size: 1em; margin: 5px 0; }
ul { padding-left: 10px; margin: 10px 0; list-styletype:disc; }
li { margin-left: 10px; padding-left: 10px; }
```

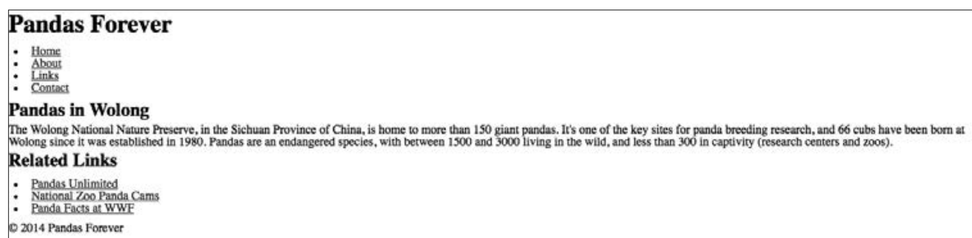


图 4-21：为示例站点中添加了内、外边距及项目符号样式后的效果

接下来，我们将为页面上的结构性元素添加一个边框，这只是为了更容易地看到元素所在的位置。即使最终的设计中是没有边框的，但在创建设计的工作进程中，临时为元素添加边框通常是有益的，这样你可以很容易地对内、外边距的值进行调整。下面是为元素添加边框的代码（你会在图 4-22 中看到结果）：

```
header, article, aside, footer { border: 1px solid #111;
}
```

为了让页面看上去更整齐一点，我们再给结构性元素加上内、外边距，结果如图 4-23 所示：

```
header, article, aside, footer { border: 1px solid #111;
padding: 10px 1em; margin: 0 5% 10px; }
header { margin-top: 10px; }
```

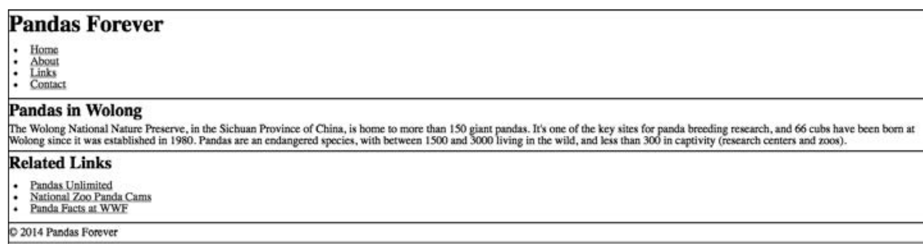


图 4-22：在示例站点中为页面的结构性元素加上边框

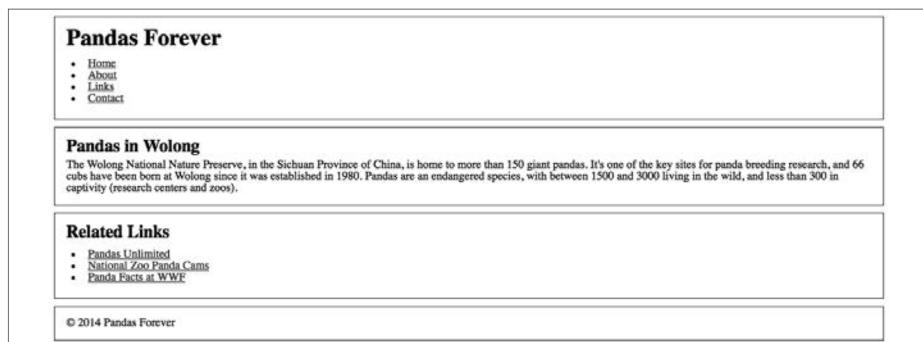


图 4-23：在示例站点中以内、外边距来分隔元素

你会发现我在添加样式到网站时没有谈及响应式设计。如果你还记得，我们一开始只有 HTML 而没有样式，这是一个响应式网站的默认做法。到目前为止，我们还没有做过任何事使其不是响应式的。该站点可以工作于不同的屏幕尺寸，如图 4-24 和图 4-25 所示，尽管在更大尺寸的屏幕上，变宽的文本使得阅读变得困难。

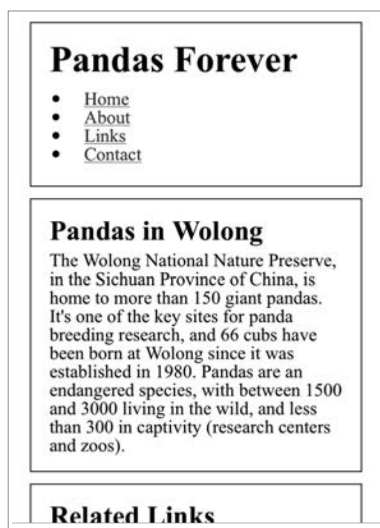


图 4-24：在窄屏幕（手机屏幕）上查看示例站点的显示效果

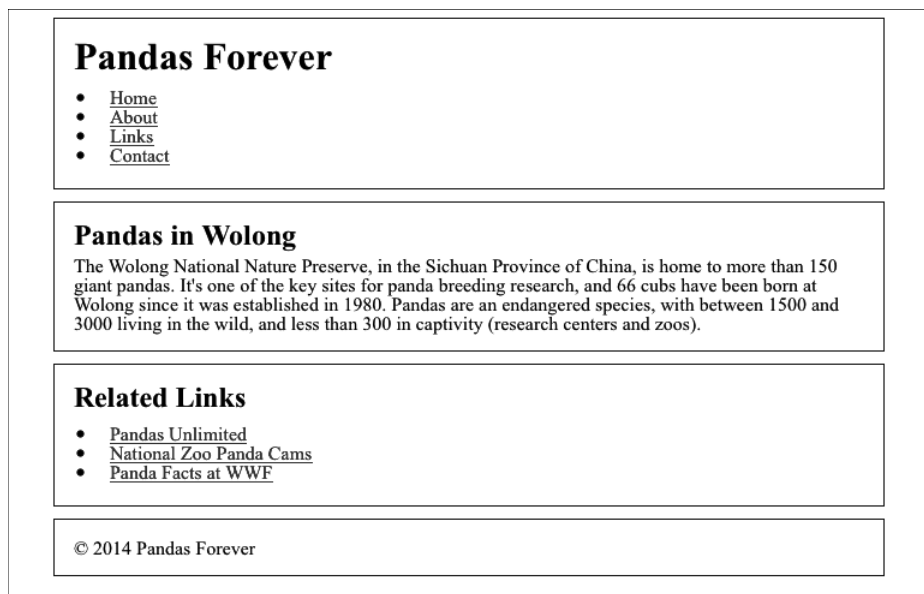


图 4-25：在中等宽度屏幕（平板电脑屏幕）上查看示例站点的显示效果

我们随后会在第 5 章中讨论媒体查询，你将学习如何改变站点布局以适应屏幕宽度。

4.13　总结

在本章中，我们知道了 CSS 是按版本发布的，与 HTML 的版本状况类似。最新版本的 CSS3 为我们带来了一些新的属性，包括响应式设计的基石——媒体查询。

尽管某些 CSS3 属性仍在测试阶段，但我们可以通过浏览器前缀在设计中包含它们，浏览器前缀使得浏览器能够渲染一个还在测试中的属性。

在网页中应用 CSS，既可通过独立的样式表文件，也可直接在网页的代码中包含嵌入样式或内联样式。一般来说，使用样式表是最好的选择，因为它允许你在全局性地应用样式于整个网站或网站的某一部分。

层叠是一套非常详尽的规则，它规定了浏览器如何在网页中应用样式，如样式以何种顺序应用，以及规则的优先级。标记为重要的规则获得最高优先级，其次是内联样式，然后是 ID、类和元素选择器，再就是继承的规则以及默认值。如果还是无法决定，浏览器将按特殊性和样式出现的顺序来决定哪条规则胜出。

虽然层叠可以是非常复杂的，但如果你在添加样式到网站时提前规划，尽可能以总体一致的方式来添加它们，就能使样式应用变得简单明了，从而只需较少的维护而不至于感到头痛。

通过重置默认值来开始你的 CSS 之旅，将确保所有的浏览器都能如你所愿地渲染样式。如果你不想自己创建，可以在网上找到各种重置样式表。

没有一种特别正确的方法来组织样式表中的代码，但最好是采用统一的方式，那样在事后也能够很容易地找到相应代码。你可以按页面或站点区块来组织样式，也可以按照样式的类别来组织，比如把文字排版样式放在一起。

CSS 布局中最重要的概念是盒模型，其决定了元素是如何在网页上显示的，每个元素表现为一个矩形盒子。每个盒子都有宽度和高度，也可能有边框、内边距和外边距。

一开始，一个元素的以上属性用的都是默认值，但这些值可以使用不同的度量单位进行修改。对于响应式设计，大多数度量单位应该是相对单位，比如百分比、em 和 rem。垂直度量单位有时可以是绝对单位，比如像素。你还可以使用 `auto`，浏览器将基于毗邻元素或包含元素来自动计算属性值。

CSS 的 `box-sizing` 属性允许你改变浏览器渲染内边距与边框的方式，也就是说，元素的宽度是否包含进了内边距与边框的宽度。定位使你能决定元素出现在页面上的位置。

现在你已经了解了 CSS 的基本工作原理，在第 5 章中，我们将学习使用 CSS 媒体查询构建响应式网站。

第5章

媒体查询

媒体查询是响应式网站能够对浏览网站的设备做出响应的关键所在。到目前为止，我们所有应用到网页的CSS声明都未考虑屏幕尺寸等设备特性。

而CSS媒体查询则允许我们基于浏览网站的设备的特性来应用不同的样式声明，最常用的设备特性是视口宽度。通过查询视口宽度，我们可以实现如下功能：站点默认为两列布局，如果屏幕（视口）宽度超过40 em，则变成三列布局。

媒体查询是响应式设计的一部分，它使得你可以在手机上看到不同于平板电脑或桌面显示器上的页面布局。媒体查询有时也称为响应式设计的秘制调料（special sauce）或魔仙粉（magic pixie dust），因为它是响应式设计中最重要的部分，没有它，也就不会有奇迹的发生。

在本章中，你将学习如何创建媒体查询并将其加入到CSS样式中。了解媒体查询可以用来查询什么（除了众所周知的视口宽度），以及浏览器对媒体查询的支持状况。

你还将学到在考虑网站的布局设计时应从最小屏幕宽度开始，再以自下而上的方式进行渐进增强。你将学习什么是断点和设计范围，以及如何在设计中使用它们。

最后，我们将回到示例站点并通过为其加入媒体查询来创建用于宽屏幕的两列式布局，并根据视口宽度的变化来改变原布局。

5.1 何为媒体查询

尽管媒体查询能在屏幕上实现非常棒的效果，可它的语法实际上相当简单。

基本上，你会以类似于“屏幕是否宽于 40 em”这样的问题起头，后带一些 CSS 样式。如果问题的答案是确定的，那么浏览器会将这些 CSS 样式应用于网页。如果答案是否定的，浏览器将忽略这些 CSS 样式。

如果你是一名程序员，你可以把它当作是一条 if/then 语句。尽管 CSS2 已经支持根据媒体类型（比如屏幕或打印机）来执行媒体查询，但直到 CSS3，我们才能够基于设备特性（比如视口宽度）来执行媒体查询。这也是响应式设计直到 2010 年才出现的原因，在这之前响应式设计不可能被实现。

这是一个简单的媒体查询例子：

```
body { background-color: green; }

@media only screen and (min-width: 40em) {
    body { background-color: blue; }
}
```

第一行的 CSS 声明将网站 <body> 元素的背景色设置成绿色。第二行 CSS 声明表示如果视口的最小宽度是 40 em，那么 <body> 元素的背景色将变成蓝色。

也就是说，窄屏获得绿色背景，宽屏获得蓝色背景。（注意：并非所有的浏览器都支持媒体查询，我们一会就将解决这个问题。）

记住，样式表中的每一条 CSS 声明会覆盖它之前（或前一个样式表中）的声明。所以，对于相同的选择器或属性，应把媒体查询放在非查询声明之后。

下面是媒体查询的一些常用示例。

- 列

  在窄屏上以一列显示，当屏幕足够宽时，变成两列、三列或更多列。

- 导航

  在窄屏上显示为隐藏导航栏（例如，下拉式菜单），在宽屏上则显示为完整的导航（可一次看到所有导航项）。

- 文字排版

  根据视口的宽度改变文字大小或调整列宽，使得每一行都能以易于阅读的最佳字数显示。

- 图片

  对于同一图片，根据页面可用空间的大小以不同的尺寸（小的特写图片与大的远景图片）显示。

> [小贴士]
> 在媒体查询中使用背景色来测试表达式是个好方法，当调整浏览器窗口大小使媒体查询生效时，你能够很容易地看到发生的变化。

5.2 媒体查询结构

媒体查询最常见的用法是直接与其他 CSS 一起包含在普通样式表中：

```
@media only screen and (min-width: 40em) {
    body { background-color: blue; }
}
```

看下这个例子，它看上去有很多代码，但很容易按含义切分成不同的部分。让我们来逐一地解释它。

```
@media
```

首先，在样式表中媒体查询总是以 `@media` 开头来表明这是一条媒体查询语句。

紧跟 `@media` 后的是一个或者多个*表达式*，它们可判别为真或假。在创建媒体查询时，必须以媒体类型（此处是 `screen`）作为第一个表达式：

```
@media only screen
```

尽管只有一个单词 `screen`，但也是一个完整的表达式，意思是“媒体类型是 `screen` 吗”。与之对应的还有 `print`（打印机）、`braille`（盲文）或者 `all`（全部）等其他类型选项，响应式网页设计的目的决定了对于媒体类型我们基本上都是用 `screen`。

你会注意到在媒体类型之前有一个额外的单词：`only`。这其实只是一个小技巧，一些老式浏览器只支持 CSS2 版本的媒体类型查询，而不支持新的 CSS3 版媒体特性查询。因此，当它们读到我们之前的媒体查询例子时，能够理解 `screen`，却理解不了 `and (min-width: 40em)`。浏览器本应该是跳过它们不理解的 CSS 声明，但对于媒体查询，浏览器的行为却有点不同，它们不是跳过整条声明，而是只跳过 `and (min-width: 40em)` 部分。结果就是老式浏览器会根据 `screen` 回答“是的，这是真的”，并忽略视口的宽度，将 CSS 应用于所有尺寸的屏幕。

我们不希望这种情况发生，所以加入了 `only` 这个单词，这实际上并不会改变媒体查询的整体含义，却能让那些老式浏览器忽略掉整个查询，而这正是我们所希望的（稍后我们将谈论如何给予这些老式浏览器合适的样式）。

[小贴士]

媒体类型 screen 涵盖了电脑和移动设备（任何有屏幕的设备），而 print 则代表打印机。

可以使用 all 来针对所有的媒体类型，但不必显式地写出来，只需简单地省略掉媒体类型即可，这将使媒体类型默认为 all。

还有一些其他的媒体类型，比如 braille（盲文）、projection（投影）、speech（语音）和 tv（电视），但它们并不常用，且也不总是如你想的那样工作。有关媒体类型的更多信息，参考 W3C 网站上的“Media types”（http://www.w3.org/TR/CSS2/media.html）。

之后是 and：

```
@media only screen and
```

需要这个单词是因为有多个表达式，screen 是第一个表达式，现在还要添加另一个表达式来评估设备特性。因为是 and 关键字，所以媒体查询中的两个表达式都需要为真，才能使整条查询的结果为真，CSS 样式也才会被应用。

媒体查询还可以使用 or 或者 not 关键字，或者有多个嵌套的表达式，但通常没有理由去做如此复杂的查询，所以在本书中我们对于这方面的内容将不做深入探讨。

接下来就到了媒体查询的主体，这是我们依据媒体特性（例如，视口或屏幕的宽度，设备的方向，或颜色性能）来判断表达式的地方：

```
@media only screen and (min-width: 40em)
```

在这里，and 之后的 (min-width: 40 em) 是你的第二个表达式（注意，表达式在括号内）。所有表达式都应包含在括号内，除非它是只有一个单词的媒体类型表达式，比如 screen 或 print。

前缀 min- 表示“至少”，也就是“大于或等于”。因此这里是说如果视口至少宽 40 em（或者如果视口大于或等于 40 em 宽）。

如果视口的宽度小于 40 em，则查询结果为假。如果宽度正好是 40 em 或者大于 40 em，则查询结果为真。

相对应的，前缀 max- 表示“最多”，即“小于或等于”的意思。

最后，用一对花括号把整条媒体查询结果为真时应用的所有 CSS 样式括起来：

```
@media only screen and (min-width: 40em) {
    ...
}
```

有一件事需要特别注意：因为你既使用花括号来包含媒体查询中的所有声明，同时也用花括号包含单条样式声明，这很容易让人迷惑。所以请确保花括号的数目正确，如果你漏掉了一个，浏览器在应用样式时将发生混乱。

在需要时，可以在媒体查询中包含多个声明：

```
@media only screen and (min-width: 40em) {
    body { background-color: blue; }
    p { padding: 5px 5%; }
    .example { color: red; }
}
```

记住，在使用媒体查询时，不要把所有样式都放入媒体查询中。正确的做法应该是先声明适用于所有视口宽度的非媒体查询样式，然后只是用媒体查询去覆盖掉那些在特定宽度中用到的样式。

所以，在本章开头的例子中，我们一开始设置所有视口宽度下的背景色均为绿色。然后在视口宽度大于等于 40 em 的时候，媒体查询以蓝色背景替代绿色背景，而其他设置保持不变，窄于 40 em 的视口还是原本的绿色：

```
body { background-color: green; }

@media only screen and (min-width: 40em) {
    body { background-color: blue; }
}
```

5.3 在样式表链接中使用媒体查询

在第 4 章中我们讲过，可以在 HTML 页面的 <head> 元素中链接一个或多个 CSS 文件：

```
<link rel="stylesheet" href="styles/mainstyles.css">
<link rel="stylesheet" href="styles/otherstyles.css">
```

浏览器从链接的第一个样式表开始，简单地读取并应用所有样式表中的所有样式声明。

不同于在样式表内编写媒体查询，我们可以告诉浏览器，整个样式表仅在媒体查询结果为真时才应用，否则就忽略掉样式表。

如果你曾经从事过网站相关工作，那么对于 print 样式表，你也许见过这样的 CSS2 版的用法：

```
<link rel="stylesheet" href="styles/mainstyles.css">
<link rel="stylesheet" href="styles/printstyles.css" media="print">
```

在这个例子中，我们告诉浏览器，如果用户是要打印页面，而不是在屏幕上浏览它，那么将加入额外的样式。比如，可以为了打印而隐藏重复的背景图像，以免浪费打印机墨水或碳粉。

要指定一个样式表仅用于打印机，只需把 media="print" 添加到 link 元素中，它告诉浏览器，当查询“媒体类型是 print 吗”为真时，就应用 printstyles.css 样式表，如果为假，则忽略之。

另一个样式表链接没有指定媒体类型，因此不管是屏幕、打印机或其他媒体类型，该样式表都会被应用。

通常我们添加打印样式来覆盖掉原样式，所以打印样式表链接应在最后列出。

告诉浏览器样式表用于打印机很简单，仅需用一个单词 print。不过我们还可以用 CSS3 媒体查询来实现这个目的，使用的语法与在样式表中嵌入媒体查询类似。

你只需以 only screen 开头，将查询放到 link 元素的 media 属性中：

```
<link rel="stylesheet" href="styles/mainstyles.css">
<link rel="stylesheet" href="styles/widerscreen.css"
media="only screen and (min-width: 40em)">
```

在这里，第一个样式表应用于所有设备，而第二个样式表仅在查询结果为真（屏幕视口宽度大于等于 40 em）时应用。

5.4 使用媒体查询的其他方式

我们刚学习了两种主要的媒体查询使用方法：一是包含媒体查询于样式表中；一是使用媒体查询链接单独的样式表。

这两种方法的最终结果是一样的，因此选择哪种方法来组织 CSS 更多的只是个人喜好问题。

此外，还有另外两种添加媒体查询到网站中的方法，同样的添加 CSS 的备选方式我们已在第 4 章中学习过。

首先，你可以在页面的 <style> 元素（<head> 元素内）的 media 属性中包含一条媒体查询：

```
<style media="only screen and (min-width: 40em)">
    ...
</style>
```

这样就意味着 <style> 元素内的所有样式仅在查询为真时应用。

我们在第 4 章中讲过应用样式于网站的单个页面通常不是一个好做法，因此可能的话也不要对媒体查询使用这种方法。

你还可以导入一个样式表，做法类似于链接一个样式表：

```
@import url(styles.css) only screen and (min-width:
40em)
```

一般来说，使用 <link> 要比导入样式表更可取。

5.5 我们可以查询什么

媒体查询能够查询许多可能的媒体特性，如视口的宽度和高度、屏幕宽度和高度、方向、宽高比和分辨率（屏幕每个维度上的像素个数）。

其中的大多数特性都可以加上 min- 或 max- 前缀。

不过，当前并非所有的这些特性都被主流浏览器所支持。我们将在本节中对其进行更细致地讨论。此外，IE8 及其更早版本不支持任何媒体查询。在本章的稍后部分，你将学习如何对此进行处理。

5.5.1 视口宽度和高度

- `width`
  视口宽度。

- `height`
  视口高度。

这是响应式设计媒体查询中最常用到的设备特性（宽度比高度更常用）。

在下面的例子中，第一条媒体查询询问视口宽度是否大于等于 40 em。第二条媒体查询询问视口高度是否正好等于 60 em（实际不太可能不用 min- 或 max-，而却要用到一个精确的度量值）：

```
@media only screen and (min-width: 40em) { ... }
@media only screen and (height: 60em) { ... }
```

在进行响应式设计时，尽管常常谈的是屏幕尺寸，但正如我在本书前面解释的那样，我们的查询实际上是基于*视口*大小，而不是*屏幕*尺寸。这也是可以通过改变浏览器窗口的宽度来查看响应式设计变化的原因。

视口是浏览器窗口中实际包含网页的那部分区域。可以看一下你的浏览器窗口，去掉页面边上的滚动条，以及顶部或底部的工具栏和菜单。

剩下的区域就是视口。

即便在同一块显示屏上，视口也会随浏览器或用户偏好（比如使用哪些工具栏）的不同而发生改变。不同浏览器侧边栏的宽度也可能会有几像素的差异，或者用户未将窗口最大化，这些都会对视口造成影响。

5.5.2 屏幕宽度与高度

- `device-width`
  设备屏幕宽度。
- `device-height`
  设备屏幕高度。

这两个特性的作用类似于视口宽度和高度，除了它们度量的是设备屏幕的实际尺寸而不是视口尺寸。因此，调整浏览器窗口大小对它们不会有影响，并且如果用户没有将窗口最大化，他可能不会获得你预期的布局。

示例：

```
@media only screen and (max-device-width: 40em) { ... }
```

5.5.3 方向

- `orientation`
  横排方向（landscape）或竖排方向（portrait）。

此特性表示的是屏幕的方向，是横排（宽度大于高度）还是竖排（高度大于宽度）。虽然这个特性可能会很有用，但还没有浏览器对其提供支持。

示例 :

```
@media only screen and (orientation: landscape) { ... }
```

5.5.4 宽高比

- `aspect-ratio`
  视口宽高比。
- `device-aspect-ratio`
  设备屏幕宽高比。

视口宽高比或设备宽高比是指宽度相对高度的比例。因此，如果一块屏幕宽 1000 像素，高 500 像素，那么设备屏幕宽高比就是 2∶1，因为 1000 是 500 的两倍。屏幕宽高比例有很

多种，尽管乍一看它们几乎都是相似的矩形外观。

常见显示器的宽高比是 16∶9（如 1920 × 1080 或 1366 × 768 像素）或者 16∶10（1280 × 800）。iPhone 3 和 iPhone 4S 是 3∶2（480 × 320 和 960 × 640），而 iPhone 5 则是 16∶9（1136 × 640）。安卓手机通常是 4∶3、3∶2、16∶10 或 16∶9。

示例：

```
@media only screen and (device-aspect-ratio: 16/9)
{ ... }
@media only screen and (min-device-aspect-ratio:
1920/1080) { ... }
```

5.5.5 分辨率

- `resolution`

  设备屏幕的分辨率。

`resolution` 有点类似于老的媒体查询类型 `device-pixel-ratio`，但已经取代它了。尽管这对在屏幕上显示恰当分辨率的图片可能是有用的，但它尚未被 Safari 和 Chrome 支持。

示例：

```
@media only screen and (min-resolution: 300dpi) { ... }
```

5.5.6 其他媒体特性

还有一些设备特性，虽然现在还不太可能会用到，但以后或许会频繁使用，包括如下这些设备特性。

- `color`

  输出设备每个颜色分量的比特位数。可用 `min-color` 对一个数（比特位数）进行查询来判断颜色性能。而只用 `color` 这个词则是询问设备是否为彩色屏幕。不被 Opera 支持。

- `color-index`

  设备颜色查找表中的条目数，比如 256。仅 Opera 支持。

- `monochrome`

  设备是否为黑白屏幕（还记得黑底绿字的老式显示器屏幕吗）。尚不被任何浏览器支持。

- `scan`

  设备是否使用逐行扫描，其只用于电视。尚不被任何浏览器支持。

- `grid`

  输出是否基于网格（如电传打字机，其字体宽度固定）或位图（一个“标准”的像素屏幕）。尚不被任何浏览器支持。

5.6 浏览器支持

所有这些媒体查询特性看上去都很不错，但却有个问题：某些浏览器根本不支持媒体查询。

因此我们需要确保在没有加入媒体查询的情况下，基本布局和设计在任何设备或屏幕尺寸上都能工作。它可能不中看，但却可用。

功能手机是不支持媒体查询的。对此就体现出了首先为小屏幕设计的一个好处，默认布局不包含媒体查询，因此能够在不支持媒体查询的小屏幕设备上工作。如果是首先为宽屏幕设计，那么默认的为宽屏幕而设计的布局在这些设备上恐怕就无法使用了。

大多数现代浏览器基本上都支持媒体查询，至少支持我们用在响应式设计中使用的 `min-width` 和 `max-width`（参见“Can I Use CSS Media Queries？”，http://caniuse.com/css-mediaqueries）。我们唯一真正需要担心的浏览器是 IE8 及其更早版本。

幸运的是，对此也是有解决方案的。是否使用该方案取决于网站用户中有多少人还在使用 IE8 或其更早版本。如果人数比例不多，给他们默认的布局设计可能就足够了。尽管它是用于小屏幕的布局（可能就一列），但一样能用在更大的屏幕上。

5.6.1 IE条件注释

如果你决定网站需要在 IE8 及其更早版本中提供对媒体查询的支持，我们可以使用条件注释（conditional comment）使代码仅针对 Internet Explorer。

这样做的缺点是会为网站增添额外的代码，而增加的多少取决于你实施的方案。

条件注释也是一种查询，但它出现在 HTML 而不是 CSS 中。你可以用它来为 IE8 及其更早版本指定一个特定的样式表。

条件注释如下所示，HTML 代码放在中间：

```
<!--[if (lt IE 9)&(!IEMobile)]>
    ...
<![endif]-->
```

用于条件注释的代码（那些括号和标点符号）有点儿难记，所以在需要时最好是直接复制它们，而不是试图强行记住如何打出这些代码。

第一行其实是说：如果 IE 浏览器版本号低于（“lt”）9，且不是用于移动设备的 IE Mobile（感叹号表示“不”），那么浏览器应该采用条件注释中的 HTML 代码。否则忽略。

整个内容在一对 HTML 注释符号（`<!-- … -->`）中开始和结束，因此所有的非 IE 浏览器会把它当作注释并忽略整个内容。而 IE 浏览器则能识别出这是一个需要估值的查询。

虽然可以使用条件注释使代码只用于 IE8 及其更早版本，但这实际上并没有加入对媒体查询的支持，那么我们如何确定将哪些代码发送给这些浏览器呢？

这实际上很好判断。因为条件注释排除了 IE Mobile（移动版 IE），那么可得知使条件注释生效的设备几乎肯定是台式机或笔记本电脑。

再考虑到常见的显示器尺寸，屏幕一般至少有 1024 像素宽，甚至是 1280 像素宽。这样我们就以此为起点，在条件注释中包含为 1280 像素 (80 em) 宽的视口准备的 CSS 样式表或样式，而不用给其一个完全不同的布局，如此也就基本能为每一个用户都提供大致准确的布局。

因此，如果要让 IE8 及其更早版本的桌面浏览器使用 `mid-size.css` 样式表，只需在网页的 `<head>` 元素中包含以下代码：

```
<!--[if (lt IE 9)&(!IEMobile)]>
    <link rel="stylesheet" href="midsize.css">
<![endif]-->
```

这里要注意的是，如果所链接的用于特定屏幕宽度的样式表中，样式是包含在媒体查询中的，这些样式将不起作用（因为条件注释实际并未加入对媒体查询的支持，你需要将样式从中提取出来）。

应对 IE8 及其更早版本浏览器的另一个办法是首先以默认的单列视图（所有媒体查询被忽略时浏览器采用的视图）开始，然后使用条件注释来加入 IE 特定布局要用到的补充样式：

```
<!--[if (lt IE 9)&(!IEMobile)]>
    <link rel="stylesheet" href="ie-styles.css">
<![endif]-->
```

[小贴士]

另一个选择是用腻子脚本为旧版本的 IE 添加媒体查询支持。比如 Respond.js（https://github.com/scottjehl/Respond），一个快速、轻量级的为 IE6-8 添加 `min-width` 和 `max-width` 支持的 JavaScript 腻子脚本。缺点是会增加额外的代码到网站中，在不支持 JavaScript 的浏览器中不起作用，且只能用于 `min-width` 和 `max-width` 查询。

5.6.2　测试媒体查询结果

想要知道浏览器回答媒体查询的结果值，你可访问由 Viljami Salminen 创建的 MQtest.io 网站（http://mqtest.io/）。如图 5-1 所示，它能精确告之浏览器发送的高度、宽度、设备高度 / 宽度、屏幕宽高比例、方向和分辨率等数值。你可以在调整浏览器窗口大小时，观察宽度和高度值的变化。如果媒体查询在某个设备上未如预期地起作用，通过这个网站你可以很方便地判定浏览器发送的信息是否与你假设的结果值相同。

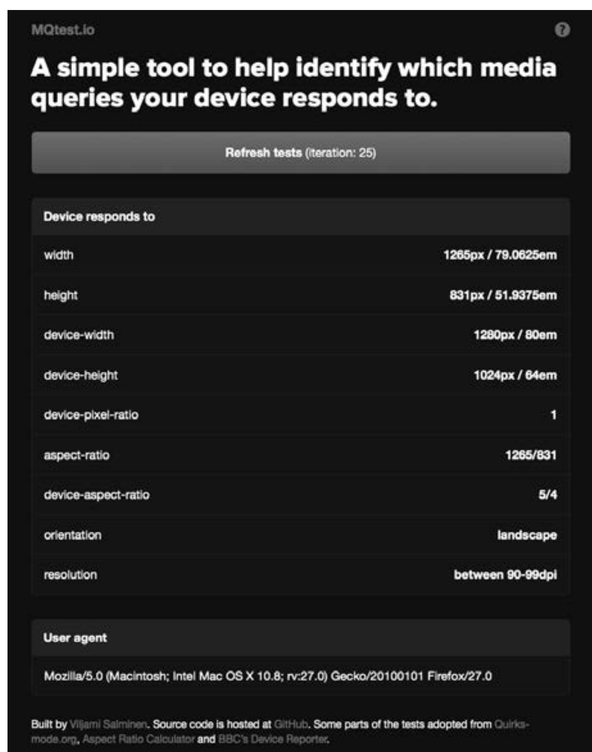


图 5-1：MQtest.io 网站将向你显示浏览器窗口的尺寸

5.7　断点

断点（breakpoint）是响应式设计中的重要概念，我们可以用媒体查询在断点处改变布局设计，使设计产生两个（或更多）的变体。

有时这会是非常明显的变化，比如列数的变化（例如，在 40 em 宽度时从一列变成两列）。有时又会是非常细微的变化，比如微调文字大小来更好地适配屏幕。

考量它的一个好办法是把所有可能的屏幕尺寸想象成一排，一端是最窄的屏幕，一端是最宽的屏幕。

在图 5-2 中，每条线代表一个断点，设计将在断点处随视口宽度而变化。



图 5–2：每条线代表一个断点

> [小贴士]
>
> 在本章的前面部分你已经学到了媒体查询可用来查询各种媒体特性——视口宽度、视口高度、方向、长宽比，甚至显示器是否是彩色的。
>
> 但因为 width 是目前大多数浏览器所支持的唯一媒体特性，它也基本上是响应式设计目前所用到的唯一媒体特性。因此，本书将只专注于使用宽度属性的媒体查询。

5.8　设计范围

在创建响应式网站时，我们趋向于将注意力集中在断点之上，它们既是我们添加到 CSS 中的一些数字，也是事物发生变化的触发点。

虽然断点十分重要，但它们只代表了一小部分视口宽度，而我们的设计应该在任何宽度下都有良好的显示效果，而不仅仅是在某些点上。

因此，就引出了设计范围（design range）的概念，也就是两个断点之间所涵盖的屏幕尺寸范围。各个设计范围获得各自不同的布局设计变体。

在下一节中，我们将添加一条媒体查询到网页中，使得页面的布局设计在视口宽度达到 36 em 时从一列变成两列，这就产生了两个设计范围：0~35 em（如图 5-3 所示）与 36 em 至无穷大（如图 5-4 所示）。而实际上我们只需针对最小 240 像素（约 15 em）和最大 3200 像素（约 200 em）的屏幕宽度做设计，因此两个设计范围最终将分别是 15~35 em 和 36~200 em。



图 5-3：页面在 15~35 em 设计范围内的一些示例

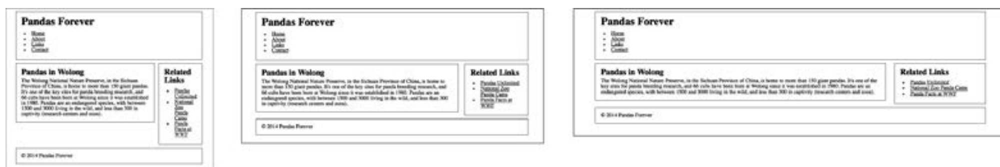


图 5-4：页面在 36~200 em 设计范围内的一些示例

在图 5-3 和图 5-4 中只给出了每个设计范围内几个点的截图，书中无法给出所有可能视口宽度下的实例页面截图。

这就是响应原型（你将在第 7 章中学习）在设计过程中如此有用的原因，你可以在浏览器中打开原型并慢慢调整浏览器窗口大小来观察设计在各个屏幕宽度下的显示效果。

当添加更多的媒体查询到示例中时，我们的设计将会被分解成更多个设计范围。

5.9　响应式设计

在向示例站点添加媒体查询之前，我们需要先讲讲构建一个响应式网站的最佳方式。

5.9.1　渐进增强

渐进增强（progressive enhancement）的理念就是从最基本的开始，然后再在此基础上为具有更强性能的浏览器和设备添加功能。

我们的默认设计（在加入媒体查询之前）针对的是最小的屏幕，同时它也需要适用于最基本的设备——那些可能无法识别媒体查询、CSS3 或 JavaScript 的设备。

在制作一个网站时，从内容入手，以 HTML 定义其语义结构，这是第一层。所有浏览器都能够阅读你的 HTML。

第二层是表现层。在表现层中，我们用 CSS 样式化页面上的内容：布局、颜色、文字排版等等。有些浏览器并不支持所有的 CSS，但没有它站点仍是可用的。但是对于几乎所有的浏览器而言，CSS 使得网站有更棒的显示效果，网站的内容更易于浏览。

第三层是行为层，是用 JavaScript 来达成的。并非所有的浏览器都支持 JavaScript，用户也可能出于安全原因而将其关闭。虽然可以使用 JavaScript 让站点做很棒的事情，比如活动菜单，但你需要确保网站的基本功能在没有 JavaScript 的情况下仍能正常使用（而这对强交互网站也许是不可能的）。

不要设置不必要的技术门槛，这只会赶跑用户。

但这也不是说就要把网站变得简单无趣，不添加任何装饰。只是说要尽可能从最简单的网站入手，并确保它能工作，然后再在这个基础上添加额外的功能组件。我们并非是“为最

低标准设计”（这是对渐进增强的一个常见批评），只是从最低标准开始，然后再为每个人进行设计。

5.9.2 用网格进行设计

现在我们暂时搁置技术问题，来谈谈如何决定内容出现在页面上的位置。

在创建一个设计的过程中会涉及很多内容，我们将在第 7 章中进一步讨论，在这里我们只是讲一下如何用网格和列来编排网页内容的基础知识。

当我们把设计整合在一起时，需要考虑设计的各个部分如何在屏幕上组合在一起。

响应式设计常被描述为包含了“灵活的网格”。 但实际上灵活性是响应式设计的内在特性，网格并不是必需的。

不过，网格确实是创建一个设计精良网站的关键。

在 Khoi Vinh 的《秩序之美——网页中的网格设计》一书中，他解释成功设计的目的是“在混沌中创造秩序”。使用网格，而不是随意创建一个又一个列，赋予了网页“秩序、连贯与和谐”，使得网页更具视觉吸引力，从而创造出更愉悦的用户体验。

用网格进行设计的想法来源于平面设计。网格意味着设计是由多个同等宽度的列组成的，列间具有相同的间距，页面上的所有内容都是基于这些列的。例如，图 5-5 是一个基于 5 列的文字排版网格。



图 5-5：一个 5 列空网格和以此为基础的布局示例

你可以看到网格有 5 列并不意味着整个设计也要被分成 5 列，网格列可以被合并来创建布局中真正的“列”。

5.9.3 使用网格列

很多网站都使用 12 列网格，因为它具有很大的灵活性：12 能被 2、3、4、6 整除，因此你的布局中可以包含使用任意这些列数的内容部分，即便是都在同一页面上。

比如图 5-6（左）所示的 Monocle 网站建立在一个 12 列的网格之上。你可以看到页面的不同部分有不同的列数，但是各部分加起来总是等于 12。例如，4 等列是 3/12 + 3/12 + 3/12 + 3/12，而 3 等列是 4/12 + 4/12 + 4/12。

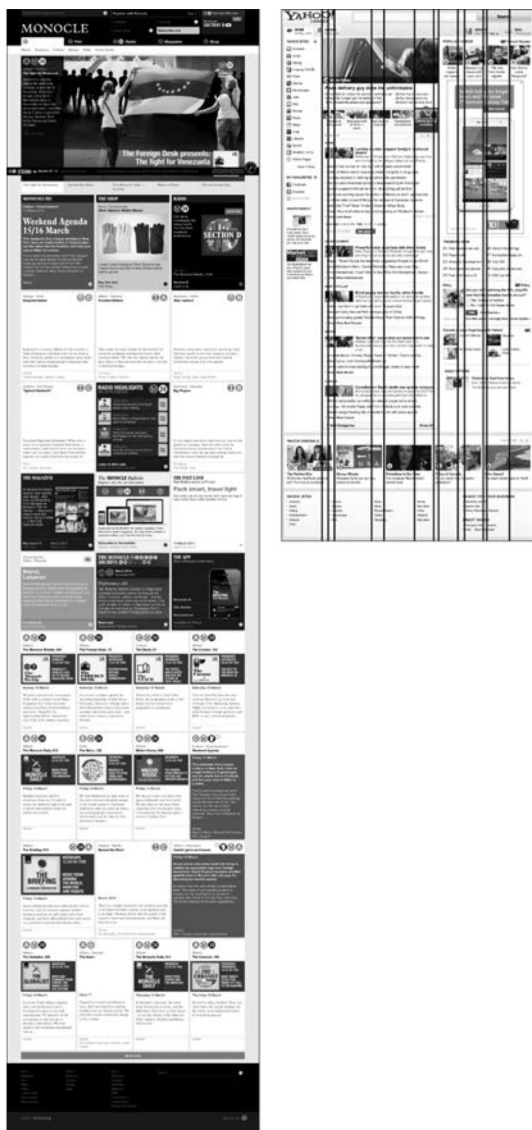


图 5-6：Monocle 网站建立在一个 12 列网格之上（左），而 Yahoo! 网站则是满页散乱的列（右）

即使向下滚动页面会看到不同的列数，但它们是连贯而有序的。

当然，在响应式站点上，这些内容块可以轻松地在不同的视口宽度下重排，以便它们适应相同的或不同的网格。

我们来对比 Monocle 和 Yahoo! 的视觉凝聚力。如图 5-6（右）所示，Yahoo! 的各内容组成部分看上去像是随机放置在一起。因此我在页面中加上线条来勾划出页面不同部分中的列数。

在本章的稍后，我们将用示例站点演示如何使用网格来决定各列应该显示在页面的什么位置。

[小贴士]

要想更多地了解如何在响应式网站中使用网格，参阅 CSS-Tricks 上 Chris Coyier 撰写的 “Don’t Overthink It Grids”（http://css-tricks.com/dont-overthink-it-grids/ ）一文。

如果你想深入研究基于网格的设计理论和方法，请阅读 Khoi Vinh 所著的《秩序之美——网页中的网格设计》一书（http://grids.subtraction.com/ ）。

5.9.4 优先为小屏幕设计

当用媒体查询基于不同的视口宽度改变网站设计时，通常会在两个彼此相对的方向中选择一个来开始工作，要么让默认设计从最小的视口宽度开始，然后为更大的宽度添加媒体查询。要么让默认设计从最大的视口宽度开始，然后添加媒体查询用于较小的宽度。

一般来说，应优先为小屏幕设计。尽管按桌面尺寸进行网站设计可能更舒服，也更容易，但从最小屏幕宽度开始设计将迫使你关注内容并只包含真正所需要的内容。

而且创建一个布局之后再将它变大要比把它变小容易得多。在你为窄视口创建了一个漂亮的布局之后，在更宽的屏幕上你就有了足够的空间去折腾内容并给它们更大的空间，但如果你尝试把一个桌面尺寸站点的所有内容硬塞进小屏幕中，那这将会变得十分困难。

如果你看下第 4 章中我们的示例站点，你会发现我们已有一个能工作于所有尺寸屏幕的布局，并在手机小屏幕上具有最佳的显示效果（图 5-7）。



图 5-7：示例站点在窄屏（手机屏幕）上的显示效果

我们将秉承这种设计并加入媒体查询以使其在所有的屏幕尺寸上都有良好的显示效果。

5.10 使用媒体查询

断点的设置是随意的。你甚至可以从零开始为网站设置任意多个断点。那么合适的起点是什么呢？最初的设计规划可以帮你大致了解需要多少断点，但你需要转到浏览器上才能实际找出添加媒体查询的最佳位置，在这个点上我们的设计将会拆分成两个设计范围。

我们先从最窄宽度开始，看看示例网站在浏览器中的显示效果，然后慢慢调宽浏览器窗口，来观察会发生什么。

5.11 两列布局

我们首先只关注文章部分和每行字符数（在第 10 章，我们将更改页面导航及其他页面组成部分）。你会在第 9 章中知晓每行文本的最佳阅读字数是 45 至 75 个字符。这并不是一个必须遵守的规则，但如果字数不在这个范围内，通常会降低阅读的舒适性。

在最窄屏幕宽度上，每行有 45 个字符，这虽处于最佳字数范围的下限，但还是不错的。

而在更大的屏幕上浏览网站时，我们希望每行字符不超过 75 个。为了找到这个点，我们慢慢调大浏览器窗口宽度，直到每行字数达到 75 个字符。如图 5-8 所示，我们需要在那添加一个断点。

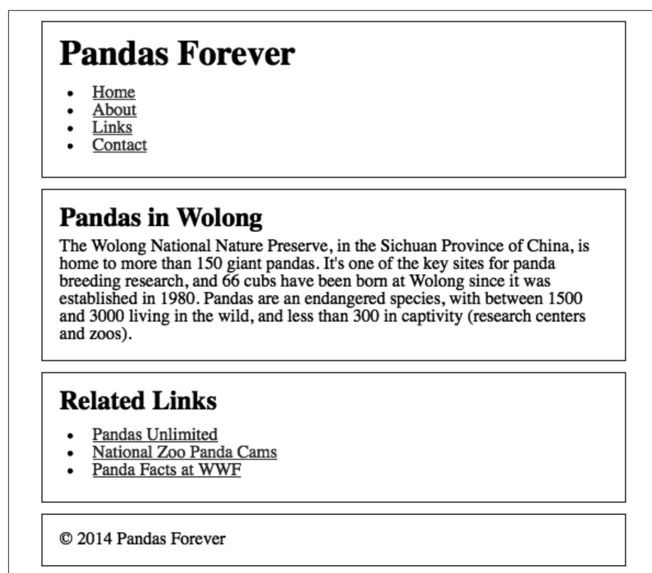


图 5-8：一旦每行超过 75 个字符，就添加一个断点

如果在那个点上测量下视口宽度，我们会发现它大约为 36 em 宽。做这件事最简单的方法就是用像 MQtest.io（http://mqtest.io/）这样的工具在同一个浏览器窗口中打开一个标签页。切换到那个标签页，MQtest.io 会告诉我们窗口当前的宽度值。

你可以将此值四舍五入，只要每行平均有 75 个字符就行，并不需要一个非常精确的值。

5.11.1　使用浮动

现在，我们想要在这个宽度上改变布局让内容分成两列显示。我们首先重温下布局所需的 CSS，然后添加一条媒体查询告诉浏览器仅在视口宽度大于等于 36 em 时才应用此 CSS。

我们把文章部分（<article> 元素）放置在较宽的一列中，并浮动（<float> 属性）至页面左边，相关链接部分（<aside> 元素）将放在较窄的那一列，并浮动至页面右边。然后为页脚添加清除浮动（<clear> 属性）样式，使其另起一行。

我们希望看到的效果如图 5-9 所示。

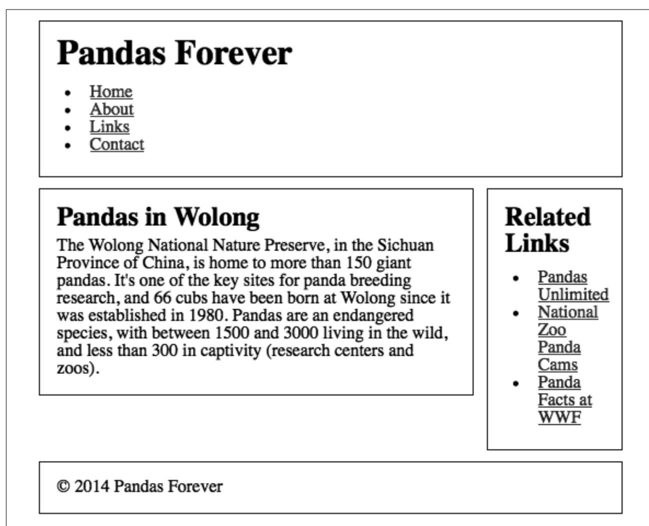


图 5-9：在这个断点处我们将使布局变成两列

所用的 CSS 样式如下：

```
article { float: left; }
aside { float: right; }
footer { clear: both; }
```

然而，当我们加上 CSS 后，会发现显示结果并不正确，如图 5-10 所示。

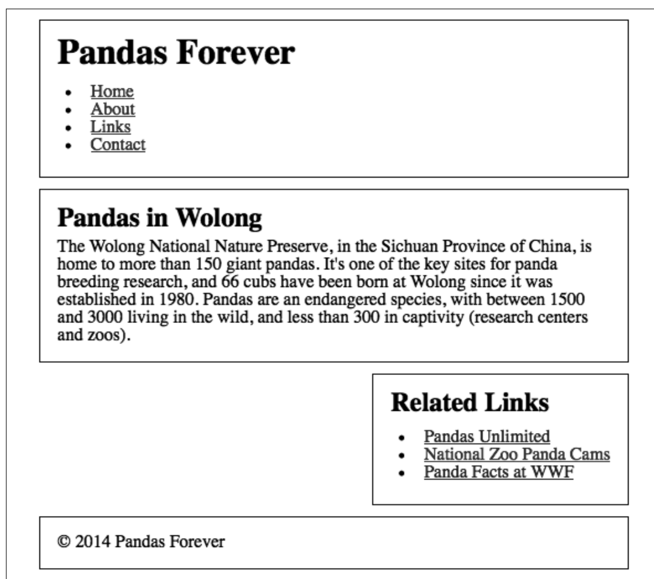


图 5-10：由于未指定宽度，文章撑满整个页面宽度，致使相关链接被挤到它的下方

问题在于，当浮动一个块级元素时，其默认宽度值变成了 auto（自动宽度），表示元素的宽度将根据其所包含的内容的需求自动调整。对于文章部分，它有足够的文本来撑满整个屏幕宽度，所以它仍和之前一样宽。然而，对于相关链接部分，其所包含内容需要的宽度很窄，因此其自动宽度也是很窄的。

5.11.2 使用网格

要解决这个问题，需要手工指定这两个元素的宽度。我们将用网格系统算出它们各自应该是多宽。

所有的宽度都是基于百分比的。我们将使用四列网格，每列的宽度是 21%，列间距是 2%，加上左右两边各 5% 的外边距。所有的数加起来正好是 100%，即整个页面的宽度。你可以在图 5-11 中看到计算出的各部分所占百分比。

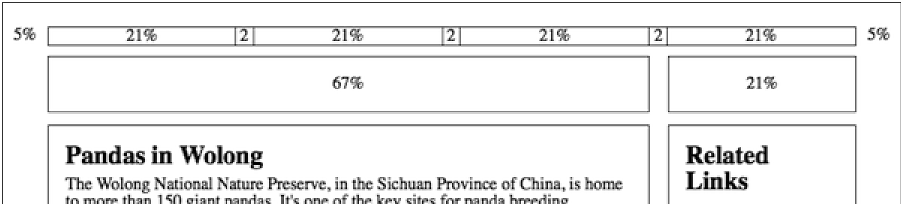


图 5-11：页面分为等宽的四列，具有相同的间距

左边的列，对应 <article> 元素，占据三列宽度。三列的宽度加上列间距的宽度合计是 67%。右边的列则为页面总宽度的 21%，两列间有 2% 的间隔。让我们添加左右两列的宽度到 CSS 中：

```
article { float: left; width: 67%; }
aside { float: right; width: 21%; }
footer { clear: both; }
```

现在距离目标越来越近了，但仍有一个问题，各列本应该是并排紧靠的，但由于两列的间隔宽度设置有误导致了如图 5-12 所示问题的发生。

现在，文章列宽度是 67%，加上左、右两边 5% 的边距。相关链接列宽度是 21%，加上左、右两边 5% 的边距。所有的数字加起来达到了 108%——这就是它们不能并排在一起的原因。两列之间的间隔应该只有 2%，而不是 10%。

那么将两列间的间隔减小就对了，因此我们把文章列的右边距和相关链接列的左边距都设置为 0：

```
article { width: 67%; float: left; margin-right: 0; }
aside { width: 21%; float: right; margin-left: 0; }
footer { clear: both; }
```

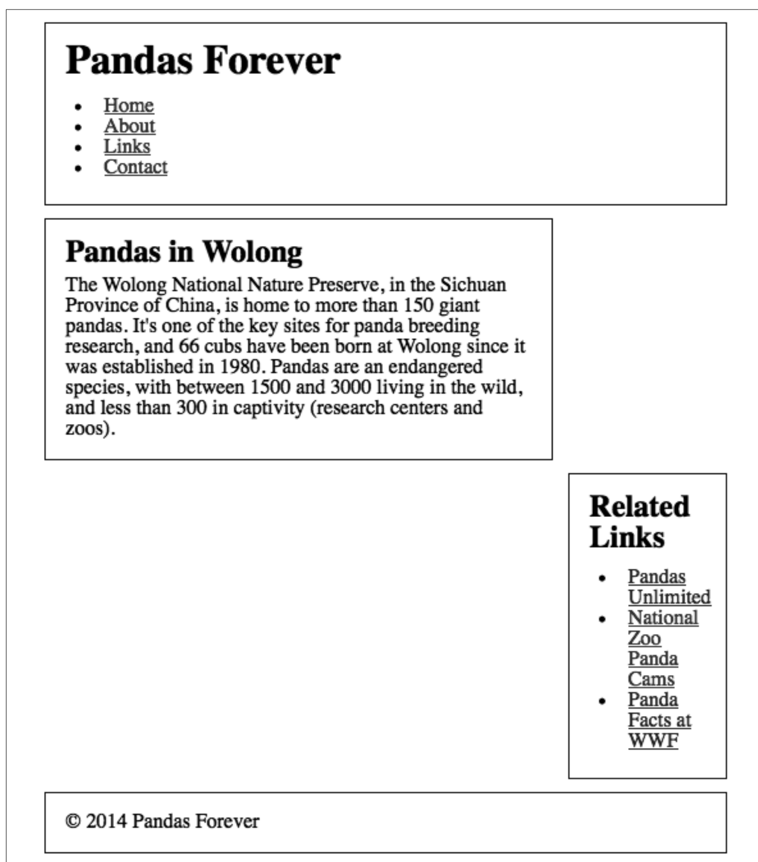


图 5-12：相关链接处于文章的下边，而正确的位置应该并排挨着文章

这样所有宽度加起来是 98%，还余下了 2% 的宽度。因为文章列总是处于左边，相关链接列总是处于右边，无需特别指明，这余下的 2% 就成了两列的间隔。现在我们获得了如图 5-13 所示的显示效果。

现在布局的显示效果已是令人满意的。如果你打算调整各列的宽度，或者它们之间的间隔宽度，只需修改下百分比值即可。

不过对于此布局，我们希望仅在视口宽度大于等于 36 em 时才应用它，所以还需要使用一条媒体查询来实现这个目标。

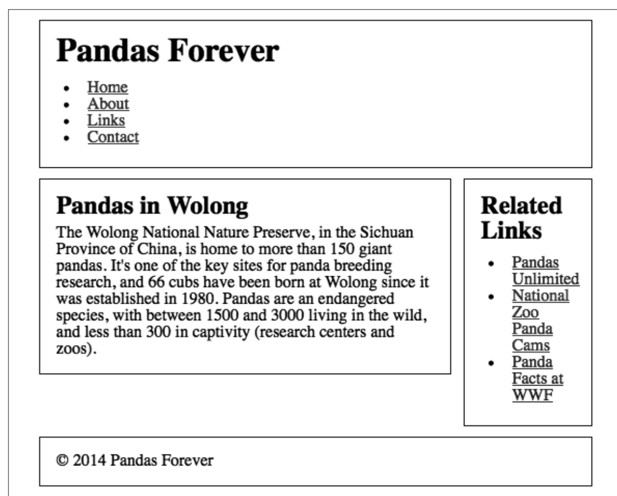


图 5-13：相关链接部分现处在了正确的位置

5.11.3 加入媒体查询

加在此处的媒体查询是非常简单的。我们希望断点发生在 36 em 时，所以要告诉浏览器，如果视口最小宽度达到 36 em，则应该应用刚加入的这三行新的 CSS 样式。

如果视口宽度大于等于 36 em，浏览器将应用这些 CSS 来生成一个两列布局。如果小于 36 em，它将忽略媒体查询中的 CSS，并保持我们初始的单列布局不变。你可以在图 5-14 中比较两个布局。以下是实现代码：

```
@media only screen and (min-width: 36em) {
    article { width: 67%; float: left; margin-right: 0; }
    aside { width: 21%; float: right; margin-left: 0; }
    footer { clear: both; }
}
```

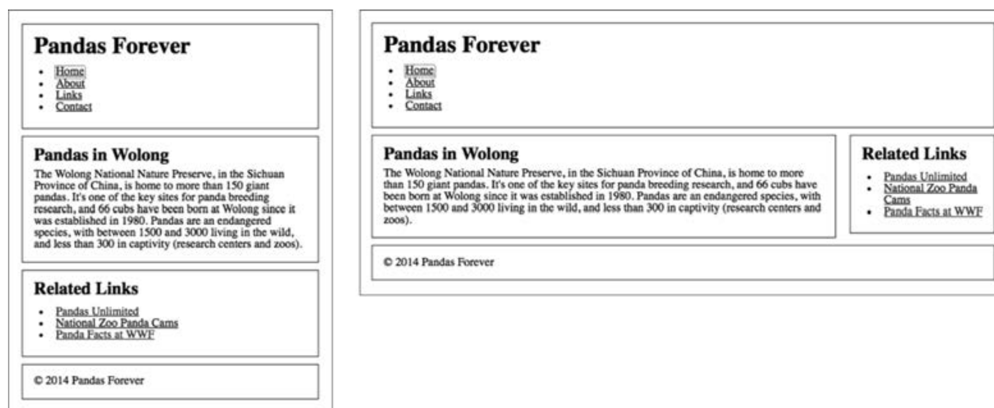


图 5-14：这是我们目前拥有的两个布局，分别用于窄于和宽于 36 em 的视口宽度

如果你要对代码进行测试，请确保已加入了此条媒体查询。当从窄到宽地调整浏览器窗口时，在 36 em 宽度处，页面将从一种布局变成另一种布局。

5.12 设置最大宽度

很好，现在我们再次使每行字数保持在了一个良好的水平，但当我们继续调宽视口，大概在 63 em 处时，每行的字数又超出了最佳范围。

对于一个标准的网站，我们可以在这个断点上添加第三列，但对于简单的示例网站，我们没有可用于第三列的内容。

那么另一种可能性就是在某个点上阻止整个页面布局继续扩大。这个设计决策可能会引发争议，因为对于非常宽的屏幕将会留下很大的空白，但在我们的例子中，页面的内容不多，所以事实上也没有其他选择了。

要限制页面布局宽度，最简单的方法就是把所有内容放入一个元素中。因此我们将添加一个 `<div>` 元素，用它包裹页面上的所有内容（它们包含在 `<body>` 标记中），并赋予 `<div>` 的 `id` 属性为 `fullpage`。

然后我们会用到 CSS 属性 `max-width`（你将在第 6 章中学到更多有关它的知识）。对响应式设计中的图片，我们会使用这个属性并设置其值为 100%，以确保显示的图片不会超出它们的包含元素。

实际上可以在任何元素上使用 `max-width` 属性，你也可以给它指定一个数值，而不是百分比。在这里，我们将使 `<div>` 元素包裹整个页面内容，并给它一个样式声明 `max-width:63 em`，这表示无论视口有多宽，这个 `<div>` 元素的宽度永远都不会超出 63 em：

```
#fullpage { max-width: 63em; }
```

果然，当我们使浏览器窗口宽于 63 em 时，布局不会再变宽。如图 5-15 所示。

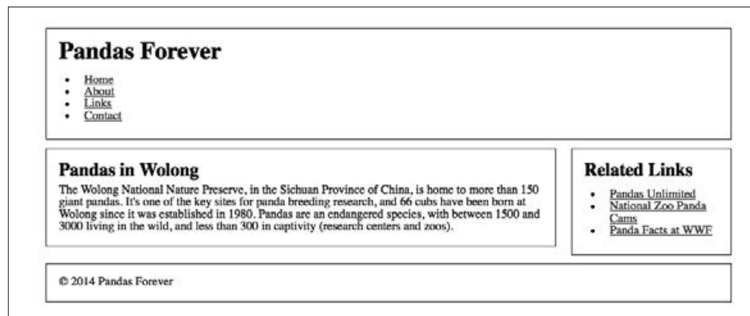


图 5-15：在宽度超出 63 em 时，页面布局不再变宽

此外还有一件事，就是我们的页面布局自始自终是与视口左边对齐的，导致视口右边一大片空白。如果能在屏幕上居中显示会更好些。要做到这一点，我们只需设置 `<div>` 元素的外边距属性值为 `auto`，这将导致布局的左右外边距平分空白区域从而使页面布局居中。如图 5-16 所示。

```
#fullpage { max-width: 63em; margin: auto; }
```

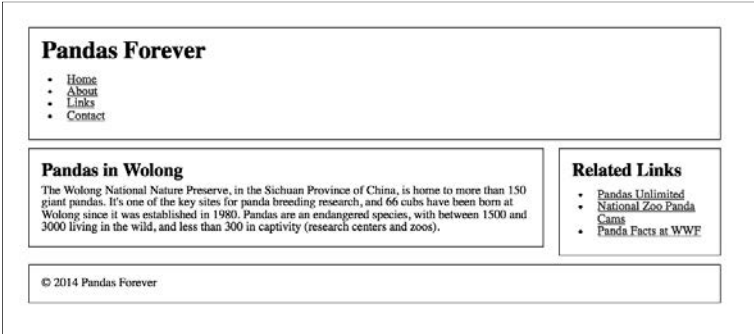


图 5-16：布局现在居中于屏幕

5.13　如何选择断点

我们已知道，可以在任一浏览器宽度上添加断点。在设计初步的草图时，你可能会对在什么位置设置断点有一个大致的想法，但在浏览器中实际测试你的 CSS 之前，是很难确切地知道需要要在何处设置断点的。在第 7 章中，我们将对设计流程中的这一步骤做更深入的讨论。

有些设计师选择让断点匹配于常见的诸如 iPhone 和 iPad 这样的设备的屏幕尺寸。但那么做只会导致你将注意力集中于每个断点处的显示效果，而你实际需要关注的是各个设计范围内的所有视口宽度下的显示效果。

当然，你尤其需要保证设计在网站访问者大量使用的那些设备上有良好的显示效果，虽然你的目标并不只是为了这些设备，而是要创建一个在所有设备上都有良好显示效果的设计（不管设备的视口宽度是什么）。

那么完全不考虑设备吗？不是的，设备仍然很重要，只是你不应该首先考虑它们。在一开始最好是不要只考虑页面设计在特定设备上的显示效果，而是应该从更通用的访问设备，比如手机尺寸、平板电脑尺寸及桌面显示器尺寸去考虑它。

在通过调整浏览器宽度观察设计变化并确定了断点之后，再在一些设备上测试显示效果。这时你可以稍微调整一下断点，因为它基于每行字符数设定，所以不必那么精确。

例如，如果 iPad 宽度与断点只差几个像素，你可以微调断点以使 iPad 落入到不同的设计范围中——比如，从单列布局变成两列布局。

记住下面的话，这真的很重要，所以我要再次重申一下——重要的是设计在每个设计范围中的显示效果，而不是在每个断点上的显示效果。

5.14 总结

媒体查询允许你基于浏览网站的设备的特性来应用不同的样式声明。这是响应式设计的关键所在，其使得依据屏幕的尺寸让用户看到不同的网站设计或布局成为可能。

尽管媒体查询功能强大，但实际使用起来却非常简单。从对设备提出询问（称作表达式）开始，如果表达式结果为真，媒体查询中的 CSS 被应用。如果表达式结果为假，媒体查询内的 CSS 将被忽略。

通过使用媒体查询，对于一个特定的样式，你一开始可以对网站应用默认值，然后再根据视口宽度用不同的值覆盖掉默认值。

媒体查询可以直接包含在样式表内，也可以用来链接一个单独的样式表，在后一种情况下，整个样式表仅在媒体查询为真时才被应用于页面。

在响应式设计中，媒体查询通常只用来查询视口宽度（即浏览器窗口中实际包含网页的那部分区域）。视口在台式机和笔记本电脑上是可变的，因为用户可以改变浏览器窗口的大小。在移动设备上，视口宽度不会变化，它与屏幕宽度相同。

还有一些其他的媒体特性也可以被查询，如方向、宽高比、分辨率，但因为这些特性并不被主流浏览器一致支持，所以它们目前并不常用。这种情况在将来可能会发生改变。视口宽度查询是被所有主流浏览器支持的，但是不包括 IE8 及其更早版本，它们需要用条件注释或腻子脚本来应用正确的样式。

断点是媒体查询中反应视口宽度的数值点，网页设计在断点上将一分为二变成两种变体。断点之间的空间称为设计范围，在此范围内的视口宽度对应一种设计变体。

为网站添加媒体查询时，先从最小的屏幕宽度开始，然后再自下而上地为更大的屏幕与设备进行渐进增强是最为容易的。此外不管屏幕是何尺寸，我们都可以用网格来增添设计的凝聚力。

在选择断点时，不应选择将断点设置为与某些常见设备宽度相同，而是应该确保常见设备宽度能够很简单地落入到某一设计范围之内。你可以在各类设备上对设计进行测试并以此来调整断点。

在下一章中，我们将学习如何在响应式网站中嵌入图片。

第 6 章

图像

到目前为止，我们已了解如何创建网页，知道如何添加基本的样式，以及如何根据视口宽度用媒体查询调整页面布局。

但是如果不在网页上放些图像，页面看起来会很单调。

你很可能听说过在响应式网站中加入图像是件多么多么复杂的事，因为图像并非总是以同样的尺寸显示。那么，如何确保不会浪费带宽向小屏幕设备发送大图像呢？

我们又如何确保发送合适的高清图像给高密度（视网膜）屏幕呢？

这是响应式设计仍待解决的一部分内容。我们将在这里讨论一些常用的解决方案，但是每种方案都存在若干缺陷，没有一种是完美的。在未来几年中这一领域很可能会出现新的思路，那也许会改变我们在响应式网站中处理图像的方式。

在此期间，我们将从本章的基本概念入手展开介绍，包括优化用于网页的图像。当对文本应用 CSS 样式可以实现相同的效果时就不要使用图像，在适当时还可使用图标字体等其他的解决方案。务必为图像设置好的替代（alt）文本，使图像对于任何人都是可访问（无障碍）的。

在决定网站上响应式图像的最佳处理方式时，你需要对比不同的可选方案并找出适合自己的方案。如果未在网站中实施响应式图像解决方案，那么网站的加载可能需要花费较长的时间。这虽然不是最理想的，但只要不是有非常多的大图像，那也并非世界末日。还有一些其他的方法可用来减少页面重量（页面显示所需要的所有文件总共的大小），我们将在第 11 章中进行讲解。

为高密度屏幕提供高清图像也是件复杂的事情，但如果做不到也没什么大不了的。使用普通图像也可以，只不过没有那么清晰。

最重要的是，你需要对响应式图像领域的新动向保持关注，从而确保能第一时间用上那些即将出现的新思路和新解决方案。

[小贴士]

在本章中我们会时常谈到图像的尺寸。在此场景下，我们使用“重量”表示图像的文件大小（通常以千字节，即 KB 计算），而使用“尺寸”指代图像的高度和宽度以避免混乱。

6.1 显示图像的方式

依据图像的用途，我们有数种方法可用来向网站中添加图像。

6.1.1 CSS替代方案

在向网站中添加图像之前，首先应考虑是否真的*需要*使用所有那些图像。

例如，使用 CSS 可以为按钮及其他元素添加边框、阴影和渐变效果（如图 6-1 所示）。直到最近，这类装饰效果还常常使用图像来实现。但现在可以使用 CSS 来获得相同的效果，这意味着网页可以下载更少的图像（也意味着更小的页面重量）。



图 6-1：用 CSS 而非图像编辑软件创建的按钮

此外，对于按钮这类元素，可使用 CSS 样式而不是用图像去装饰，此后要想在全站范围内改变颜色及其他样式就很方便，只需改下 CSS，而不必重建每个图像。CSS 样式化的元素也更具响应性，改变它们的大小并不会降低其显示质量。

并不是所有的老式浏览器都支持这些 CSS 样式属性，但使用这些浏览器的小部分用户仍将获得可用的按钮，只不过是可能没有圆角或阴影效果。

创建类似按钮效果的代码比较复杂，但如果在网上搜索“CSS 按钮生成器”（CSS button generator）你会找到很多的网站，它们让你可以通过组合搭配所有可能的 CSS 属性值来设计一个按钮，然后为你生成所需的 CSS 代码，供你复制并粘贴到自己的样式表中。

你还可以在网站上使用 CSS 而不是用图像实现艺术字效果。几乎所有可以在图像编辑软件

中对文字所做的事，比如轮廓文字、垂直侧转文字等效果在 CSS 中都可以实现。利用这一点，你可以确保样式化文本是响应式的同时也是可访问（无障碍）的。

不过，对于像公司标志（logo）这样的事物，需要总是具有完全相同的显示效果，那最好还是使用图像以确保它的所有细节都能正确地显示出来。

6.1.2 内容图像

内容图像是那些向用户传达意义的图像，它们要么是网站信息的一部分（比如报纸文章配图），要么作为导航元素（比如引导用户访问网站社交媒体页面的图标）。

要记住一点，用户并非都以相同的方式浏览网站。某些用户使用的设备可能无法显示最新版 CSS，而另一些用户使用的是屏幕阅读器，它会用替代文本代替图像显示。

对于属于网站内容部分的图像，你通常会用 HTML 的 `<img>` 元素。记住，对于结构化的 HTML 页面，每个元素都应有其意义。

`<img>` 元素是为数不多的只有一个标记，不需开始 / 结束标记对的元素。它也有很多属性：

```
<img src="images/norwaypine.jpg" alt="An evergreen tree
with big bushy branches.">
```

总是要用到的两个属性分别是 `src` 属性和 `alt` 属性，前者告诉浏览器去哪里找到图像文件；后者在浏览器无法显示图像时提供一个替代的文本。我们很快将会更深入地学习替代文本。

6.1.3 背景图像

CSS 的 `background-image` 属性允许你添加背景图像用于装饰页面，它们不成为内容的一部分：

```
p { background-image: url(flowers.png); }
```

不要使用 `background-image` 属性添加内容图像，因为它们对于使用屏幕阅读器或者其他文本浏览器的人们来说是不可见的。

在使用 CSS 的 `background-image` 属性为元素添加背景图时，图像将作为元素的背景显示在它所包含内容（比如段落中的文本）的后面。在本章的后面部分我们会给出一个背景图示例。

6.1.4 图像拼合

我们将在第 11 章中了解到，网页会为它所需的每个文件（CSS 文件、图像等）都生成一个单独的 HTTP 请求，这将增加页面的加载时间。减少请求的数量能够加快页面的加载速度。

我们可以通过把许多小图像合并成一个大图像来减少需要加载的图像文件数量，然后通过 CSS 定位技术只将大图像中的特定部分显示在页面的正确位置上。

这种合并图像的技术称为图像拼合（image sprite）。

例如，图 6-2 是苹果公司用来显示其网站顶部导航栏的的拼合图像，包括每个按钮的各种状态（活动、悬停等），都拼合在一张图像中。

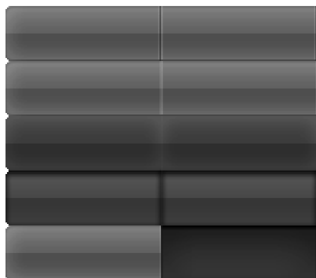


图 6-2：用于显示苹果网站导航栏诸多状态变化的拼合图像

实际的导航效果如图 6-3 所示，对于当前活动页面（图中是“iPhone”），其背景与其他按钮不同。拼合图像中的每个矩形切片提供一种可能的按钮背景，通过 CSS，每个按钮每种状态下只能显示一个特定的矩形切片。



图 6-3：用拼合图像创造出的一种版本的导航栏效果图

当然，正如我们在本章前面学到的一样，类似这样的按钮通常可以用纯 CSS 来实现，根本不需要图像！

6.1.5 图标字体

图标字体是一个优秀的小图形解决方案。图标字体集跟其他字体类似，只不过字母数字字符被符号图形所替代。图标字体示例请见图 6-4。

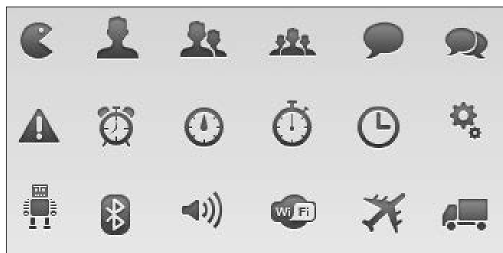


图 6-4：出自 Yummygum 的 IconSweets 图标字体（http://iconsweets2.com/）

图标字体与标准字体一样都是矢量字体，即每个字符无论是以怎样的大小显示都不会失真。这意味着它们具有良好的伸缩性，即使是在大尺寸和高密度显示屏上。

虽然每个字符只能是一种颜色，而非多色，但你可以选择任何颜色，就像为文本选择颜色一样（也可以像对文本一样使用你喜欢的其他 CSS 来样式化它们）。

你会找到一个差不多包含了你所需的所有符号标志的图标字体。通常图标字体是基于主题的。比如有一个图标字体集包含了世界上所有国家的地形轮廓符号。你还可以找到天气图标、社交媒体图标、动物、传统符号等任何的图标。

使用 `@font-face` 来在设计中包含图标字体，就像使用其他的字体一样（你将在第 9 章中学习如何给网站添加字体）。

Chris Coyier 发表在 CSS-Tricks 上的文章“Using Fonts for Icons”（http://css-tricks.com/using-fonts-for-icons/）说明了使用图标字体的好处。同时也请你务必读下 Drew Wilson 刊登于 Pictos 上的“Using Icon Fonts”（http://pictos.cc/articles/using-icon-fonts/）一文以弄清如何确保图标字体是可访问的。

你可以在网上搜寻图标字体（很多是免费的），甚至还有网站允许你创建自己的图标字体。

尽管图标字体文件通常都不大，但应合理使用它们，避免滥用。

6.2 替代文本

你要考虑到不是每个访问你网站的用户都能够*看见*网站。

大多数情况下这指的是盲人和使用屏幕阅读器的用户。他们看不到图像，所以屏幕阅读器会读出替代文本（alt text）来为用户描述图像的内容。你需要设置合适的替代文本以便这些用户不会错失内容中的重要部分。

而且也不是只有盲人用户是你的替代文本的潜在受众。有时图像由于网络连接或文件问题而无法加载，在这种情况下，用户将在屏幕上本应是图像出现的地方看到你的替代文本。

在图 6-5 中 , 你看到的是所有图像都正常显示的美国铁路公司网站，而在图 6-6 中所有的图像都被替代文本所取代。



图 6-5：美国铁路公司网站，所有图像正常显示

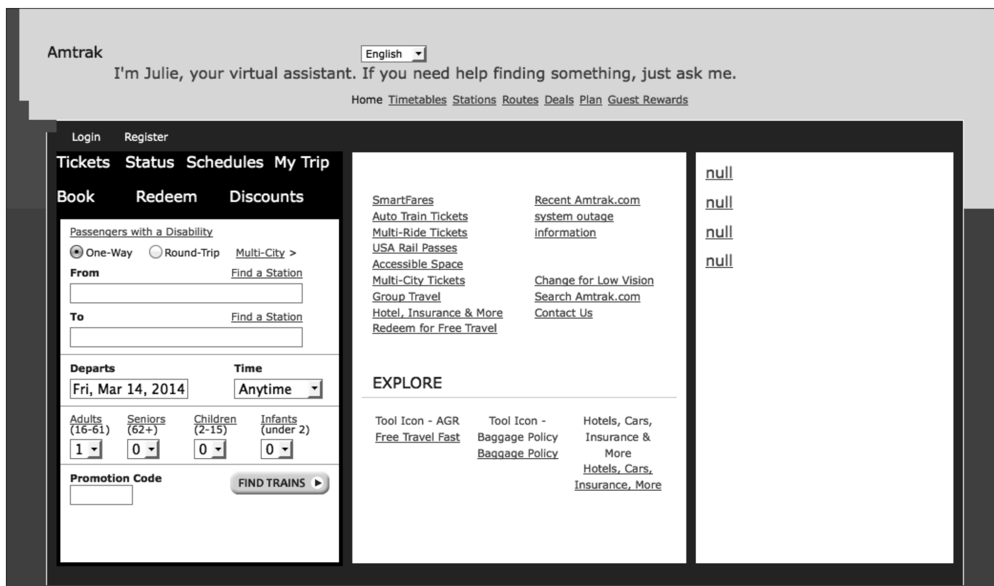


图 6-6：美国铁路公司网站，所有图像被替代文本取代

你会注意到，内容图像被替代文本所取代，比如左上方的美国铁路公司标志。不过，标志旁边的列车长头像没被替换，因为它是装饰背景图像，对内容无足轻重。

如果用替代文本取代图像并无意义，可以使用空值作为 `alt` 属性的值：

```
<img src="images/decorative.jpg" alt="">
```

不要轻易省略 `alt` 属性。如果这样做了，屏幕阅读器将无法得知图像对内容是否有意义，并且因为没有替代文本，作为第二选择，阅读器将朗读文件名给用户。

使用空值将告诉屏幕阅读器完全跳过图像。

美国铁路公司网站上的一个问题是，右列中的促销图像没有指定替代文本，甚至连空值都未设置，所以它们显示为“null”。

编写好的替代文本

仅仅是为图像添加了替代文本还不够，需要让文本真正告诉用户在内容场景中图像包含了什么。替代文本是对图像所提供信息的替换品，而不仅仅是图像的描述，我将解释这其中的差别。

要弄清楚对于一个特定的图像其替代文本应该是什么，想象你正在为他人大声读网页。当你遇到图像时，你会怎么说？

当然，图像标题也提供了图像的信息，所以请记住你不需要再重复这一信息。

例如，考虑图 6-7 中的图像。假设图像的标题是“沿着佛罗里达 101 号公路”。



图 6-7：根据围绕内容的不同，一张图像可以有很多种解释

要编写出好的替代文本，你首先需要从内容所在的场景考虑。

例如，可能这幅图像是一条关于飓风的新闻报道的配图。表达的是飓风对棕榈树的影响。你的替代文本可以是“暴风雨中棕榈树的树枝全被刮向一个方向”。

或者图像是昨日一场经过佛罗里达的飓风的新闻报道的配图，传达的信息是描述飓风经过

时公路上的情况。你的替代文本可能就变成了“空荡潮湿的公路旁棕榈树倾斜在风中”。

同一幅图像伴随不同的内容有不同的含义，你不可能描述出图像的每一处细微差别，所以你需要决定什么样的替代文本是与上下文最相关的。

尽管可能只有少数用户会看替代文本，但它对这些用户是非常必要的，你应该为此花心思，而不只是随便给图像设置个如“树木”这样简单的替代文本。虽然这也对图像做出了描述，但它没有为用户提供任何有用的信息。

加入尽可能多的细节描述对于充分传达图像的含义是必要的。但是，也应控制替代文本的长度。通常保持在 256 个字符以内（包括空格）比较合适，当然了，实际上对于替代文本并没有字数上的限制。

你并不总是需要对图像的视觉外观进行描述。例如，一个公司的标志通常可以用公司名及公司标志上的附加文字所代替。你不需要描述样式化后文本的颜色和外观。

在其他情况下，图像可能会显示冗余信息。假设你有一个页面列出一项活动的赞助商，每一个列表项中公司名（文字）旁边都有公司的标志图像。如果你设置标志图像的替代文本为公司名，那意味着在屏幕阅读器中列表项将被读作“公司名，公司名”，在这种情况下不为标志设置替代文本更好，这样就只会读一次“公司名”。

如果你不确定所编写替代文本是否合适，只要想象一下页面中没有图像，看看你的替代文本是否对页面有帮助意义。浏览器或浏览器插件通常有这样一个选项，可以用替代文本代替页面上的图像，那样你可以清楚地看到效果怎样。

不要用“XX 的图像”或“XX 的照片”这样的文本打头来设置替代文本，屏幕阅读器在读替代文本之前，会告之读的是图像。

使用 `title` 属性提供有关图像的附加信息是一个相当普遍的做法，当用户移动鼠标光标到图像上时会“弹出”一个信息框显示附加信息。但这却不是一个好的做法，因为其意味着信息并非对所有用户都是可用的（移动设备用户、只有键盘的用户、大多数屏幕阅读器用户）。如果你有信息要提供给用户，应包含在图像标题中，它是对所有用户都可见的。

不要只图简单，而直接复制替代文本到 `title` 属性中，因为有些屏幕阅读器两者都会读。更多的信息，请查看 Steve Faulkner 在 The Paciello Group 网站上撰写的“Using the HTML title attribute”（http://blog.paciellogroup.com/2013/01/using-the-html-title-attribute-updated/）一文。

6.3 图像文件格式

一般来说，你用在网站上的图像应该是 JPEG、GIF、PNG 格式的。所有浏览器都可以显示这些文件格式。我们也会讲一下 SVG 文件格式，它比其他类型的文件更具灵活性，但目

前并不是所有的浏览器都支持 SVG 格式的图像文件。通过文件扩展名我们可以知晓图像文件的格式类型。

文件格式之间的差异对于响应式设计有重要的影响，因为你选择的格式会影响到图像的文件大小。你应该使用既能够产生最小字节数，同时图像的效果仍符合你预期的文件格式。

还有一些其他的图像文件类型，但它们无法工作于所有的浏览器，所以应该避免使用它们。如果一个你要用在页面上的图像的格式与常用格式不同，使用图像编辑软件将它转换成 JPEG、GIF 或 PNG 格式的文件。

6.3.1 JPEG

文件扩展名：.jpg 或 .jpeg（读作“jay-peg”）。

这是传统意义上最常见的一种用在网页中的图像格式。适合于照片及其他包含很多种颜色的图像。

6.3.2 GIF

文件扩展名：.gif（读作“gif”或“jif”）。

这种类型的图像通过减少文件中使用的颜色数量来压缩文件大小（产生较小的文件）。一个 GIF 图像最多可显示 256 种的颜色，这意味着它一般不能用于可能有成千上万种颜色的照片。图 6-8 显示了一张照片保存为 GIF 格式后的效果。它看上去是失真的，因为没有足够的颜色来融合所有的像素。



图 6-8：因为颜色数不够，这张 GIF 图像看上去是失真的（另见彩插图 6-8）

因此，GIF 格式最好用于颜色范围较小的图像，比如标志或插图，而不是照片。

不同于 JPEG 图像，GIF 图像中可以有透明的区域，这类 GIF 图像显示在页面上时，背景会透过图像的透明区域显示出来。

此外，GIF 图像能够保存多个不同的帧（按顺序逐一显示），可用于显示非常短的动画剪辑。虽然这可能偶尔会用上，但文件大小却会大大增加。你可以使用 CSS 来重现此动画效果，其对网站性能的影响要少得多。

另外要考虑到的是很多人觉得网页上的动态物件会分散注意力，所以最好避免使用动画图像。

6.3.3 PNG

文件扩展名：.png（读作“ping”）。

有两种类型的 PNG 图像，当在图像编辑软件中以 PNG 格式保存文件时，会被问到以哪种类型的 PNG 格式来保存图像：

- 8 位 PNG 文件只支持 256 种颜色，类似 GIF 文件，但因为压缩算法的差异，同样的图像 PNG 格式的文件可能要小于 GIF 格式的文件；
- 24 位 PNG 文件支持约 1600 万种颜色，类似 JPG 文件，但因为压缩算法的差异，同样的图像 PNG 格式的文件通常要大于 JPG 格式的文件。

两种类型的 PNG 文件都像 GIF 文件一样支持透明。不过，24 位 PNG 文件还支持半透明，你可以看到图像后面的整个背景（网页），而并不只是图像空白区域后的那一部分（IE6 及其更早版本不支持此效果）。我们也可以使用 CSS 的 `opacity` 属性来实现类似的效果。

6.3.4 SVG

文件扩展名：.svg（读作“ess-vee-gee”）。

SVG 是一种矢量图形格式。相比其他格式保存的是图像所有像素点的数据，SVG 则是以 XML 格式保存图像信息，其中定义了图像各部分（线条、形状、颜色、渐变、滤镜等）及各部分之间的关系。这种类型的图像文件只适用于图形，不适用于照片。

由于 SVG 是以这种方式而非逐像素存储数据，所以它与分辨率无关。你调整图像至任意尺寸都不会降低图像质量，所以不需要针对不同的分辨率创建单独的图像。

唯一的缺点是 IE8（及其更早版本）和 Android 2.3（及其更早版本）均不支持 SVG 图像，因此在这些浏览器中，你需要使用 JavaScript 以其他格式来代替此格式的图像。你可以用诸如 SVG Swap（https://github.com/teleject/svg-swap）之类的插件实现此功能。

想要更多地了解使用 SVG 图像的相关信息，可以去读下 Chris Coyie 发表在 CSS-Tricks 上的文章“Using SVG”（http://css-tricks.com/using-svg/）。

6.4 优化图像

在网站中，影响用户体验的一个最大因素就是性能——完成页面加载和渲染需要花费多长时间。如果时间过长，用户会放弃并离开。更糟糕的是，在一个慢速连接上无论用户等多久，页面都没法完成加载。

性能瓶颈最常见的原因之一就是图像。

6.4.1 像素与分辨率

在讨论优化图像之前，我们需要讲一下像素。你可能知道像素是什么。在电脑屏幕上，像素是一个彩色（或灰度）的物理点（屏幕可以显示的最小的点）。你在屏幕上看到的图像实际上是由一个个点组成的，与你在电视上或打印在纸上看到的图像是一样的。有那么多的点以至于我们通常无法分辨出每个单独的点，但放大之后，它们看起来如图 6-9 所示。

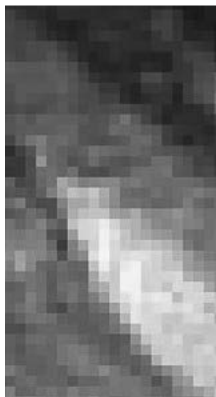


图 6-9：屏幕上一幅图像中的单个像素放大后的样子（另见彩插图 6-9）

像素的数量决定了电脑屏幕的分辨率。例如，1024 × 768 分辨率的显示器屏幕是 1024 像素宽乘以 768 像素高，这通常被认为是一台 15 英寸显示器（12 英寸宽乘以 9 英寸高，即 15 英寸对角线长度）。

直到最近为止，像素的大小还总是相同的。你从屏幕分辨率就可看出屏幕有多大。1280 × 1024 分辨率的屏幕在物理上肯定大于 1024 × 768 分辨率的屏幕。

这就是我们设计的内容以像素为单位的原因，因为它们在所有的屏幕上总是相同的大小。一个 300 像素 ×300 像素的图像在屏幕上总是以相同的大小显现，不管是用什么显示器查看它。

但近年来一些设备制造商意识到，如果他们将像素变得更小，就能在屏幕上塞入更多的像素，就会使屏幕上的内容看起来更“真实”。

例如，以物理尺寸来说，iPhone 的屏幕似乎应该是 320 像素 × 480 像素，但它的实际分辨率是 640 像素 × 960 像素，在每个方向上都多出了两倍的像素。

这对于我们刚才提到的 300 像素的图像意味着什么？当 iPhone 屏幕在以实际分辨率显示网页时，会*表现得好像*屏幕是 320 像素 × 480 像素一样。要不然，图像上的一切都将显得实在是太小了。因此，300 像素的图像在每个方向都将是用两倍之多的像素显示：不是 300 像素 × 300 像素，而是用 600 像素 × 600 像素。

所以现在我们需要区分 iPhone 在显示网页时它自称所拥有的像素（320 像素 × 480 像素）和实际构成屏幕的物理像素（640 像素 × 960 像素）之间的差别。我们在设计中参考的是它自称所拥有的像素，称为*参照像素*或 *CSS 像素*，而屏幕上的物理像素称为*设备像素*或*硬件像素*。

6.4.2 高密度屏幕

这些新的如 iPhone 这样的设备像素多于参照像素的屏幕称为*高密度屏幕*（high-density screen）。根据定义，这些屏幕的*像素比率*（pixel ratio）大于 2。多出的像素使得屏幕的*显示效果*要优于普通屏幕，因为具有的像素足够多，人眼不能够分辨出单个像素，而且它看起来更加的“真实”。

苹果公司称这种屏幕为*视网膜屏*，你可以在新的 iPhone、iPad 和 MacBook Pro 上见到它们。举例来说，iPhone 3 的分辨率是 320 × 480，iPhone 4 拥有相同的物理尺寸，但分辨率却是 640 × 960，在每个方向上的像素数都是原先的两倍（合计则是 4 倍之多）。

设备像素和参照像素之间的差异可能会非常混乱，这也是我们在网页设计中转向使用如 em 和百分比之样的相对单位的一个原因，那样我们就不必管屏幕是如何定义像素的（这可能在将来的设备中会发生变化），而使用相对度量单位，将使设计在每个屏幕上都有相同的显示效果。

不同设备以不同的方式呈现像素，而且这一切都还在逐步变化中。

如果你想了解更多这方面的信息。参阅 Scott Kellum 发表于 *A List Apart* 上的文章“A Pixel Identity Crisis”（http://alistapart.com/article/a-pixel-identity-crisis）。

高密度屏幕的问题在于，如果发送普通图像给这些屏幕，屏幕将以预期的尺寸显示图像，但因为它们填充空间所用的物理像素数是普通屏幕的 4 倍之多，这通常会导致图像显示失真或模糊。

最佳的解决方案是提供一个两倍尺寸的图像。例如，用 400 × 400 的图像替代 200 × 200 的图像，但显示在相同的 200 × 200（参照像素）空间中。在高密度屏幕上，图像看上去是相同的尺寸，但会显得非常清晰。

但既然我们想为高密度屏幕提供不同的图像，那我们将会有文件体积更大，需要的加载时间更长的图像。你不会希望普通屏幕用户浪费带宽去下载这些大图像。我们将在后面谈及响应式图像——如何基于屏幕是否为高密度屏来使用不同的图像时解决这个问题。

如果你使用的是两个一组的对应图像，一般约定用于高密度屏幕的图像其文件名末尾加上“@2X”。例如，你的文件可能会被命名为 tree.jpg 及 tree@2x.jpg。

用于高密度屏幕的图像并不一定总是原图像的 2 倍大小，因为高密度屏幕可以有各种像素密度。有时为不同分辨率的屏幕同时创建 1.5 倍和 2 倍的图像是可取的，尽管这可能并不一定必要，而且有可能只会使问题变得更加复杂。

我们可以使用包括 JavaScript 或 CSS 在内的各种解决方案来切换图像，使高密度的屏幕获得高密度的图像。要了解更多信息，请阅读 Edward Cant 发表在 Menacing Cloud 上的博客文章“Optimising for High Pixel Density Displays”（http://menacingcloud.com/?c=highPixelDensityDisplays）。

如果没有能力或资源来为网站添加单独的高密度图像，我们有两个选择。

首先，你可以对所有屏幕都使用高密度版本的图像，只要确保图像尺寸正确。这样做的缺点是图像会很大，需要更长的时间来加载。如果你的网站中图像不是很多，这是一个可以接受的解决方案。

第二个选择是坚持使用常规密度版本的图像。它们仍将显示在高密度屏幕上，只是可能会略有模糊。查看网站在高密度屏幕上的显示效果，以决定你是否可以接受。

当然，你可以混合使用这两种方案，一种作为通用方案应用于网站中的大多数图像，另一种则只应用于网站中的特定图像，不必对所有的图像都采用一种处理方案。

6.4.3 压缩图像

对图像进行压缩能够减小文件大小，并有助于更快地加载图像。

在图像编辑软件中，你可以找到用高、中、低品质保存图像的选项（查找像“存储为 Web 所用格式”或“导出”这样的选项）。尝试以不同级别的压缩品质来保存图像，看看文件大小的变化情况。有些图像在较低的质量下会显得模糊或失真，而无法用在网站上。但有些图像设置成较低质量仍有较好的显示效果，从而使得你可以使用更小的文件。

改变图像文件格式也可以减小文件大小。试试分别以 JPEG、GIF 或 PNG 格式保存图像，看看文件大小是否有差别。再次重申，务必保证你的选择不会明显降低图像质量。

Photoshop 等应用程序提供有选项可以在保存之前在屏幕上并排查看图像的不同版本，并可比较文件的大小和类型。

在显示效果仍然不错的前提下，尽量使用那个占用空间最小的图像版本。请务必在不同的设备屏幕上对其进行测试，因为在不同的屏幕上可能会有不同的显示效果。

也可以试试来自雅虎开发者网络的 Smush.it 在线工具（http://www.smushit.com/ysmush.it/），它可以在不改变显示质量的情况下优化图像。

要了解更多有关如何为网络优化图像的知识，阅读 Christopher Schmitt 的《响应式 Web 图形设计》（*Designing Web & Mobile Graphics*：*Fundamental Concepts for Web and Interactive Projects*，由 New Riders 出版）。

[小贴士]

即使你对图像进行压缩优化以用于网站，你仍需保留原始的高分辨率图像，以便不时之需。追踪原始文件的窍门是将它们与优化后的图像上传到网站的同一个目录中。即便高分辨率图像并不会在网站中实际使用，但如果有人在将来需要用到它们，他们会知道可以在哪里找到这些原始文件。

现如今存储的价格已很便宜，所以保存它们的成本极小。给予原始文件的文件名要能清楚表明其是原始文件，比如 flowers_original.psd，那样你就很好分辨它们了。

6.4.4 实际尺寸

如果未使用 HTML 或 CSS 告诉浏览器图像的尺寸，浏览器默认会以全尺寸来显示图像。所谓全尺寸也就是你在图像编辑软件中保存图像时的图像尺寸，比如说 300 像素 × 400 像素。

有时以全尺寸显示图像没什么问题，但有时基于视口宽度，你可能希望图像比实际尺寸小一些。我们可以用 CSS 来使图像变小。

不过，你不会希望图像大于其实际尺寸显示，因为这最终会使图像变得模糊不清。幸运的是，浏览器决不会这么做，除非是故意用 CSS 这么做。图 6-10 和图 6-11 比较了实际尺寸图像和以大于其实际尺寸显示的图像之间的差异。

对于响应式设计，图像尺寸可能会因屏幕宽度的不同而要进行调整，因此为能适应所有可能的屏幕宽度及保证图像的显示质量，一开始应选用最大尺寸（在所有视口宽度下）的图像文件。

但是对于一个特定的屏幕尺寸，你当然不会希望网站去下载比实际所需尺寸大得多的图像文件。我们将在本章稍后部分对这个问题进行处理。



图 6-10：以实际尺寸显示的图像（另见彩插图 6-10）



图 6-11：大于实际尺寸显示的图像（另见彩插图 6-11）

6.5 内容图像

当添加内容图像到网站中时，我们要使用 <img> 元素。在接下来的几节中，我们将介绍如何将内容图像添加到网页中，放置在正确的位置上，并确保它们是正确的尺寸。

6.5.1 <img>元素

<img> 是一个行内元素。在下面的代码中，<img> 元素在 <h2> 和 <p> 两个块级元素之间，因此图像也堆叠在这两个元素之间，如图 6-12 所示：

```
<h2>Pandas in Wolong</h2>

<img src="images/pandaphoto.jpg" alt="A panda eating bamboo.">

<p>The Wolong National Nature Reserve, in the Sichuan
Province of China ...
```

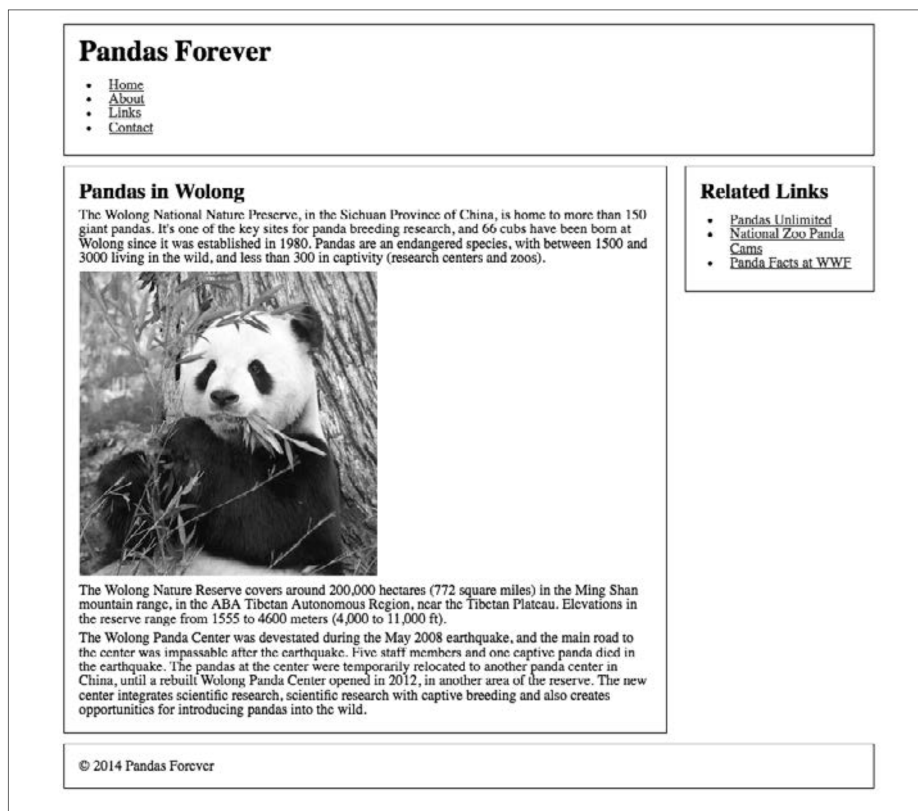


图 6-12：网站的主内容块中加入一张熊猫图像

与迄今为止我们所见过的其他元素略有不同，<img> 元素不像标题和段落元素那样在其包

含的内容之间有开始 / 结束标记，相反，它只有一个标记，并使用 `src` 和 `alt` 属性来决定显示在页面上的图像。这两个属性都是必需的。

`src` 属性包含图像文件的链接网址，如果图像文件与网页在同一网站时，可以是相对链接（images/pandaphoto.jpg）；如果是其他网站上图像，则应是绝对链接（http://www.example.com/pandaphoto.jpg.）。

`alt` 属性我们已经讨论过，它告诉浏览器在图像不可用时显示什么内容。

6.5.2　加入图像

现在让我们看下页面布局在最大视口宽度下的显示效果。是不是觉得需要对照片旁边的那一大块空白区域做些什么？

你也许在很多网站、杂志和报纸上见过它们通常使文字环绕在图像周围以填充那些空白区域。这将用到 `float` 属性，我们在第 4 章中已讨论过它。

我们将给予图像 `float： left` 样式，那样元素将始终放置在左边，后面的内容将围绕它“浮动”以填充其四周可用的空白区域，如图 6-13 所示：

```
article img { float: left; }
```



图 6-13：照片在主内容区中浮动至左边

也可以浮动图像至右边，如图 6-14 所示：

```
article img { float: right; }
```

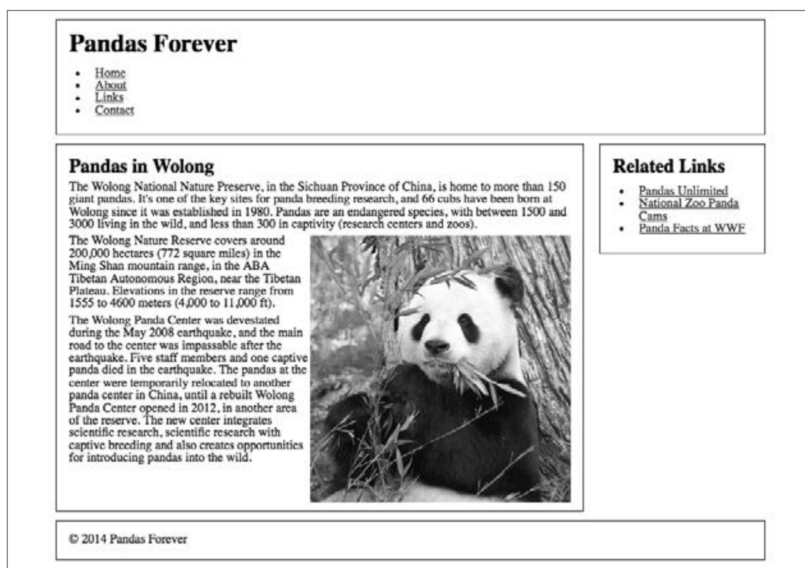


图 6-14：照片在主内容区中浮动至右边

然后可能还需要加上点内边距，这样看起来效果会更好，如图 6-15 所示：

```
article img { float: right; padding-left: 3%; paddingbottom: 10px; }
```

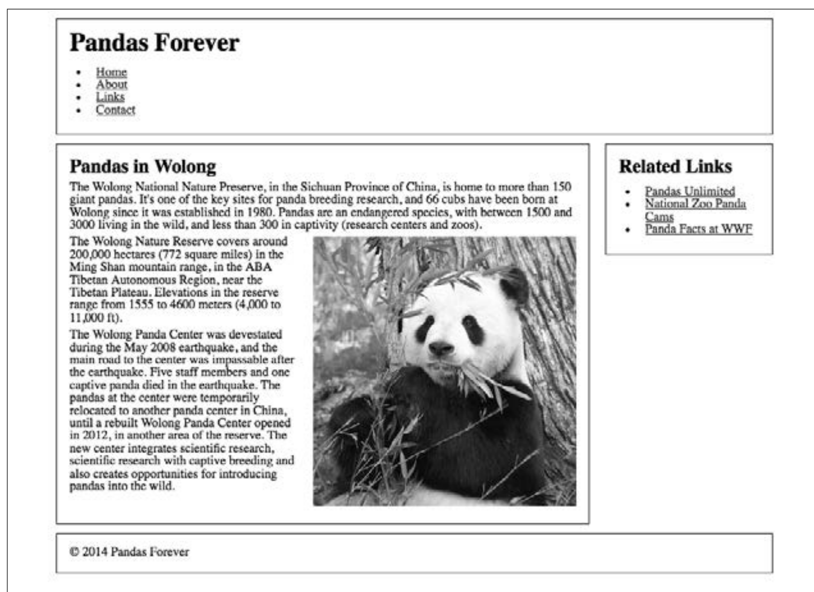


图 6-15：给浮动图像四周添加内边距

6.5.3 灵活的图像尺寸

以上图像是以全尺寸显示的，因为我们没有使用 CSS 来改变其尺寸。在最大页面宽度上，这看起来很好——图像足够大，可以看到很多细节，也有足够的空间容下围绕它的文本。

但是如果把浏览器窗口变窄，则会觉得图像周围的文本有些拥挤，如图 6-16 所示。

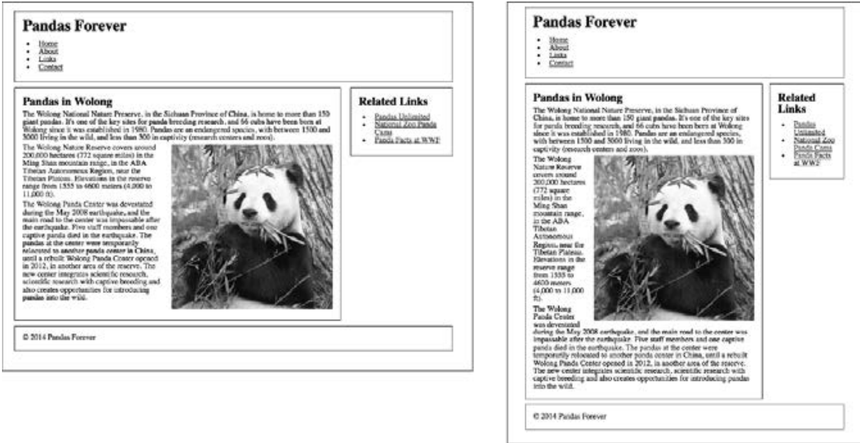


图 6-16：在较宽的页面中图像有足够的空间，但当屏幕变窄时，它将变得拥挤

当视口变小时，图像跟着变小是明智的，因此不需要占用那么大的屏幕区域。我们可以将图像宽度设置成 50% 来做到这一点，这意味着图像的宽度是其包含元素宽度的一半。

```
article img { width: 50%; }
```

如图 6-17 所示，你可以看到图像在各种宽度的屏幕上其尺寸都是合适的，直到宽度窄至 36 em 时布局将切换成单列布局（我们很快会看到）。

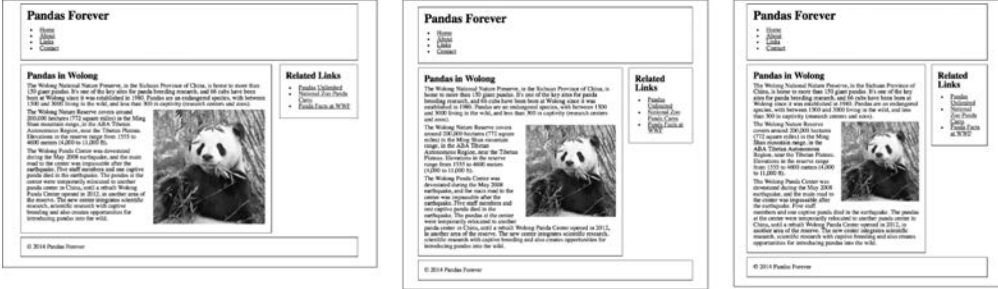


图 6-17：在屏幕宽度变窄时，图像也随之变小

因此设置了 CSS 之后，在较大的屏幕上，你可以利用充裕的空间以较大的尺寸来显示图像。而在较小的屏幕上，图像仍是一个适宜的尺寸，不会挤压其他内容。

[小贴士]

我们可以通过 HTML 精确设置图像的宽度和高度，这在过去曾经是常用的图像设置方法（例如，`<img src="image.jpg" alt="alt text" width="400" height="200">`）。但是，如果使用 HTML 设置图像的尺寸，就会丧失响应性。因此，我们应避免这样做。如果你需要指定图像尺寸，用 CSS 来设置。

6.5.4 媒体查询

当屏幕变窄直至 36 em 断点处时，页面变成单列布局。图像占据了单列布局的一半宽度，在 36 em 及比之略小些的宽度中，这仍是一个适宜的尺寸。

但当你继续减小视口的宽度，如图 6-18，会发现图像实在是变得太小了，以至无法看清细节。这是你就会觉得它应该更大些才好。



图 6-18：屏幕宽度更窄时，图像变得太小以至无法看清细节

修正这个问题很简单，我们只需添加一条媒体查询，在此视口宽度（或更窄）下图像将占据整列宽度，而不只是一半。我们需要在 28 em 处添加一个断点。

这一次，我们是按从宽至窄的屏幕（视口）顺序设置图像，所以在媒体查询中要用 `max-width` 取代 `min-width`。

我们将图像宽度设置为 `auto`，这使得图像将以其实际尺寸（300 像素宽）显示。为覆盖我

们先前已经添加的样式，我们需要在样式中加入 `float: none` 使其不再向右浮动，并去除我们在有文字环绕图像四周时使用的左内边距：

```
@media only screen and (max-width: 28em) {
    article img { width: auto; float: none; padding-left:0; }
}
```

现在我们获得的显示效果如图 6-19 所示。

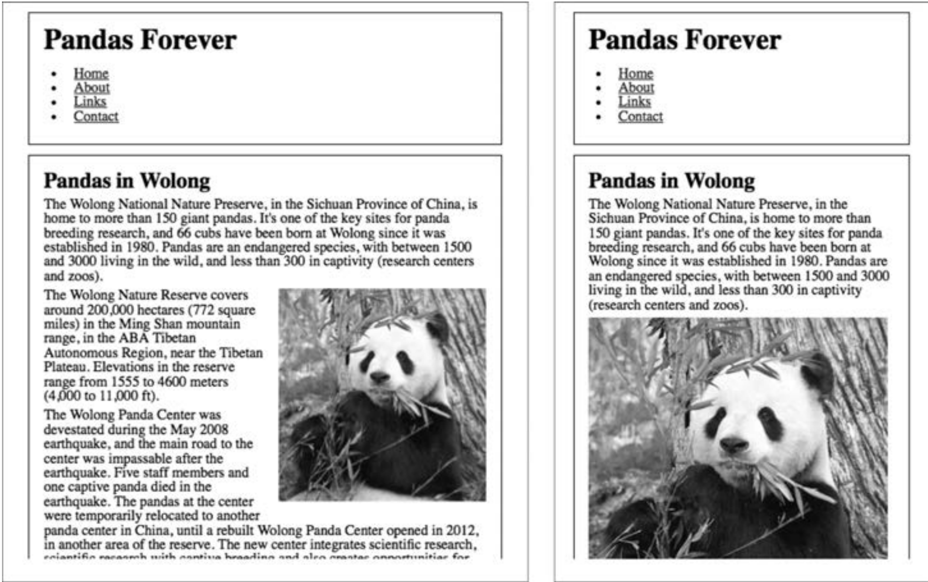


图 6-19：对于窄屏幕，图像将占满整个屏幕宽度且不再有文字环绕其四周

你可能会奇怪，为什么是按从宽至窄的方向添加图像媒体查询，而不是像先前那样从最小屏幕（视口）宽度开始。事实上我们可以选择从任何一个方向开始。具体到为网站添加图像，有时按由宽至窄的方向要更加容易，因为图像尺寸在其实际宽度上是有硬性限制，不可能无限制扩大（在本例中是 300 像素）。

当然，在样式表中混合使用 `min-width` 和 `max-width` 媒体查询也是可以。问题在于，对这张熊猫图像我们的默认样式（没有任何媒体查询）是适用于较宽屏幕的，所以如果小屏幕不支持媒体查询的话，那得到的就是并不适用于它的宽屏幕图像。

但不用担心，即便是同时从两个方向添加媒体查询，我们仍可以轻松地翻转其中的一些媒体查询，使它们转为相同的方向（不过如果网站中的所有媒体查询都是按一个方向定义这往往能减少混乱）。

例如，我们在前面例子中使用的这些样式默认是适用于较宽屏幕的，然后再用媒体查询添加用于较窄屏幕的样式：

```
article img { float: right; padding-left: 3%; paddingbottom:10px; }
@media only screen and (max-width: 28em) {
    article img { width: auto; float: none; padding-left:0; }
}
```

翻转过来则成了这样：

```
article img { }
@media only screen and (min-width: 28em) {
    article img { float: right; padding-left: 3%;
    padding-bottom: 10px; }
}
```

不过且慢，为什么翻转后默认用于较窄屏幕的样式，在第一行中对于 `article img` 没有应用任何样式？这是因为在翻转前的例子中所有用于较窄屏幕的样式（`width: auto; float: none; padding-left: 0`）是用来将 CSS 属性恢复成默认值的。

而在翻转之后，样式本身就都是默认值，所以我们不需要再显式地声明它们。（当然，退一步讲，整个 `article img { }` 行也是不必要的，因为其中没有定义额外的样式。）

6.5.5 最大宽度

在网站上使用 `max-width` 是使响应式设计正确工作的关键之一，它实际上超级简单。下面举个例子来说明它是如何工作的。

我们的图像在 28 em 断点处看起来还不错，当然我们需要看下整个设计范围，下移至最窄宽度 320 像素。遗憾的是，问题产生了，见图 6-20。

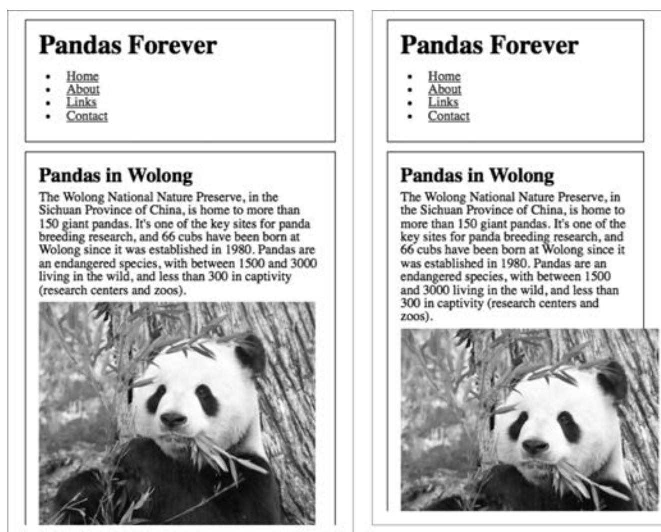


图 6-20：在这个宽度下，图像超出了其容器宽度

一旦我们缩窄窗口至 22 em 左右时，图像的宽度超过了列的宽度！不仅是超出了列的边界，图像右边也因为超出了浏览器窗口宽度而被切断。

幸运的是，这个问题很容易解决。

我们肯定不想只简单地给图像设置一个百度比宽度，因为我们想要它尽可能大，直到 300 像素的实际宽度。

相应的，我们可以使用一个非常方便的属性：`max-width`。你应该记得在第 5 章中我们使用这个属性告诉我们的两列布局在达到某一视口宽度后不再继续变宽。在那个例子中，我们设置宽度值用的是 em 单位。

对于这幅图像，我们将设置 `max-width` 值为 100%。通过使用 100%，我们告诉浏览器图像的显示宽度决不能大于它的包含元素的宽度，在本例中，就是决不能超过 `<article>` 元素的宽度：

```
article img { max-width: 100%; }
```

在图 6-21 中，最左边的图显示的是 28 em 宽度下的页面布局，我们可以看到尽管图像是以其实际尺寸显示的，但并没有完全将整列填满，在我们将窗口变得更窄时，图像也随之变窄来与列宽匹配（它仍将呆在包含元素内部决不会超出）。

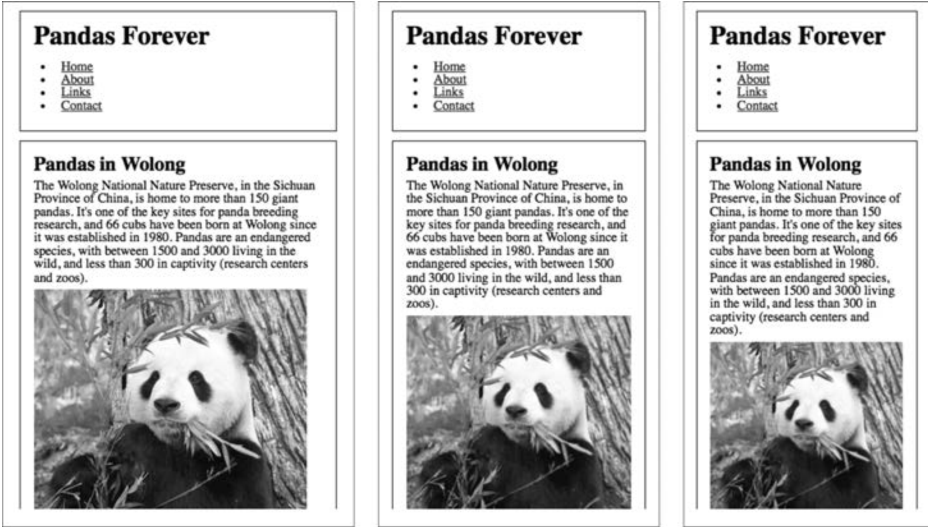


图 6-21：使用 **`max-width:100%`**，图像绝不会宽于其容器

如果你研读过很多关于响应式设计的资料，你可能已经知道 `max-width` 是响应式设计的三大要素（流体栅格、弹性图像和媒介查询）的一部分。它确实至关重要，没有 `max-width`，响应式网站将无法工作，因为你会经常遇到图像无法如预料那般适配的问题。

这里有一个小秘密：对于你的响应式网站，因为 max-width 对于图像是如此重要，可以简单地将它应用于整个网站。那就是在样式表中添加如下代码（把它放在重置代码下面就好）：

```
img { max-width: 100%; }
```

一旦你这样做了，就不必再考虑它了，就这么简单！

尽管你并不需要对网站中的所有图像应用 max-width:100% 样式，但其并无任何害处，因为它做的唯一的事情就是确保图像不会宽过它们的容器元素。它不影响其他与图像大小有关的事物。但在极少的情况下，我们可能不需要将 max-width:100% 应用于一个特定的图像，那么对该图像简单地添加 CSS 来覆盖默认设置就可以了。

[小贴士]

在 IE7 及其更早的版本的浏览器中使用 max-width 会产生一些问题，比如按比例缩小的图像显示效果不好。如果你需要支持这些浏览器，查看 Ethan Marcotte 在博客文章所介绍的一种 JavaScript 解决方案“Fluid Images”(http://unstoppablerobotninja.com/entry/fluid-images/)。

6.5.6　用图像叙事

在选取放到网站上的图像时，务必挑选有趣并能真实传达网站个性的图像。

使用购买的素材图像是一个极具诱惑力的主意，而且这之中有很多照片确实很棒，但通常它们并未经过精心挑选且看起来很做作，使人一眼就能看出它们是素材图像，这让你的网站看上去缺乏个性。

如果你有一个餐厅网站，不要使用某人在吃汉堡之类的素材图像。应该用你自己餐厅及美食的真实照片，并确保照片的高质量，为此去聘请专业的摄影师是值得的。

最重要的是，尽可能多地使用标题。照片的故事并不总是能从其本身就能明显看出来。繁忙的餐厅的照片很好，但是有一个解读标题会更有效“繁忙的周六晚上，餐厅总是早早就坐满了剧场观众，为避免等待请记得预约”。

6.6　背景图像

并非所有图像都是内容的一部分。你也可以有装饰性的图像，这能增加网站的设计感，但不会增加任何意义。

你可以使用 CSS 的 background-image 属性来添加属于样式部分而不是内容部分的图像。

不要用 background-image 添加内容图像，因为使用屏幕阅读器或其他文本浏览器的人们是无法看到它们的。（一个例外：实际上有方法可以显示一个 CSS 图像然后用 HTML 提供隐藏

于可见网站中的替代文本，但这种方法不正规且困难，几乎不可能成为一个好的设计选择。）

6.6.1 加入背景图像

比方说，我们要在示例网站的页眉部分添加背景图像。

我们将再次从最大屏幕宽度开始。在我们的最大宽度设计中，排除边距，页眉大约是 900 像素宽，所以我们需要使横幅图像至少也有那么宽。页眉的高度大约是 140 像素高，但根据字体大小的变化它能变得更高，所以我们需要使横幅图像足够高以使页眉能肯定被覆盖（见图 6-22）。



图 6-22：背景图（另见彩插图 6-22）

为页眉添加背景图像很简单。你可以在图 6-23 中看到结果：

```
header { background-image: url(images/pandabanner.png); }
```

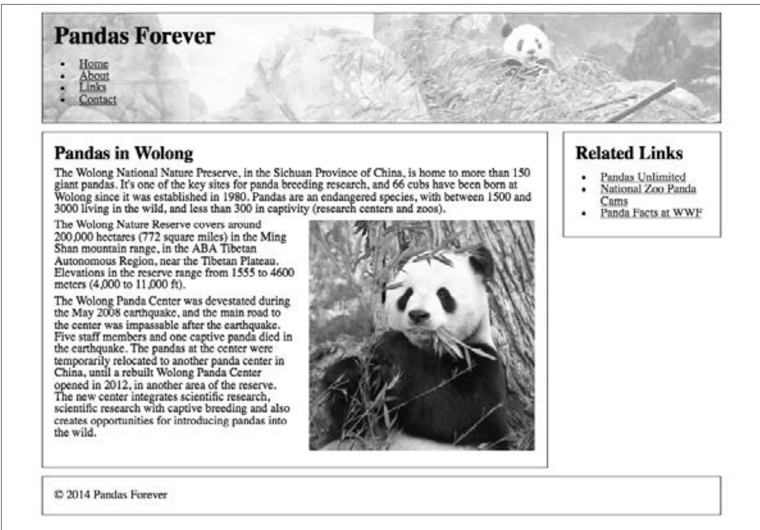


图 6-23：背景图像加入到页眉中

你会注意到，横幅图像的亮度要比实际的照片浅。这就是所谓的透明度（transparency）或不透明度（opacity）。在本例中，我想要照片浅一些，这样就不会影响阅读它上面的文字。

你可以使用 CSS 的 `opacity` 属性来调整内容图像的不透明度，但要在背景图像上这么操作可不那么容易。因此我先在图像编辑软件中改变图像的透明度，然后再上传到网站中。

6.6.2 对齐

如果你把浏览器窗口变窄，看看横幅图像的显示，你会注意到原来多少还算是在横幅图像中央的熊猫现在偏向了右边，如图 6-24 所示。



图 6-24：对于较窄的屏幕宽度，横幅图像不再是居中的

这是因为，横幅图像是左对齐（默认）的，图像的最右边因为在元素中容不下而被切断了。

不像内容图像，背景图像是作为样式应用于元素的，所以它永远不会超出元素的边界（你根本看不到图像的超出部分）。因此，没有必要设置 `max-width`。

默认情况下，背景图像对齐于左上角。可以使用 `background-position` 属性连同两个值一道在水平方向（沿 *x* 轴）和垂直方向（沿 *y* 轴）上对齐图像。

在我们的例子中，我们既要居中图像，又要保持它在顶部，如图 6-25 所示。我们可以像下面这样做：

```
header { background-image: url(images/pandabanner.png);
background-position: center top; }
```



图 6-25：可以使用背景定位使图像居中

你也可以使用百分比或固定的度量单位（比如像素）来对齐背景图像。

6.7 响应式图像

在响应式网页设计中一个主要的讨论话题是响应式图像。我们如何为大屏幕提供相应尺寸的图像以充分利用屏幕空间，但又不浪费带宽发送这些更大的图像给小屏幕设备呢？我们如何提供清晰的图像给高分辨率屏幕，而不是把这些大文件发送给不需要它们的只有低分辨率屏幕的设备？

如果我们有多个尺寸的图像文件用于不同的屏幕，我们要确保设备仅下载它们需要的那个图像，而不是全部。

人们提出了大量可能的解决方案，我们在这里讲述一些最流行的解决方案，但没有一个是完美的。

在未来的几年，在该问题上很可能会有更多的讨论，也绝对有可能产生其他响应式图像解决方案。

现有解决方案的一个主要问题是，它们全都太复杂。很多网页内容并非是由开发人员创建的，人们使用内容管理系统（CMS）来编辑自己的网站。所见即所得（WYSIWYG，What You See Is What You Get）的编辑器让网站编辑通过点击按钮并选择文件即可添加图像。例如，在一篇博客文章中添加一张照片。尽管这其中的一些解决方案可以集成到 CMS 中，但网站编辑们几乎无法干涉去选择最佳的图像尺寸（也可能不具这样做的知识）。

即使对于开发者来说，如果一种响应式图像技术过于复杂，那很多人就会回避它，并继续使用简单直接的 <img> 元素。

创建多个图像来切换也不是未来友好型的。我们基于今天的屏幕大小来选择图像尺寸和分辨率，但它们在将来可能迥然不同。

6.7.1　建议的客户端解决方案

理想的情况下，应该有一种方法基于设备性能，使用被所有浏览器支持的 HTML/CSS 来切换图像。

这是 W3C 响应式图像社团（http://responsiveimages.org）正在努力达成的目标。

不幸的是，此项工作还在进行中。有两种可能的响应式图像解决方案，<picture> 元素和 srcset 属性，但这两者没有被任何一个主流的浏览器实现，并且也还不是 HTML 规范的一部分。

这两种解决方案或全部或其中一个很可能成为将来的 HTML 规范的一部分，所以我们在这简略地介绍它们，你可以了解它们是如何工作的（尽管你还并不能在你的网站上使用 <picture> 元素或 srcset 属性）。

在本章稍后的内容中，我们将探讨当前可使用的其他解决方案。

1. srcset

加入到 <img> 元素的 srcset 属性允许你指定并上传同一图像的多个版本（不同的文件），浏览器将会只下载它需要的那个文件，而不是所有文件。

目前，<img> 元素的 src 属性允许你指定一个图像文件，而 srcset 属性将允许你指定多个文件（以逗号分隔）。每一个图像文件将根据它适用的视口宽度或像素密度进行标记。然后浏览器将选取最合适的图像，下载并显示它。

让我们来看个例子：

```
<img src="images/flower.jpg" alt="a flower" srcset=
"images/flower-HD.jpg 2x, images/flower-small.jpg 600w,
images/flower-small-HD.jpg 600w 2x">
```

在本例中，默认的图像是 flower.jpg。而高密度屏幕（像素密度大于等于 2）则使用 flower-HD.jpg。对于小于 600 像素宽的窄屏幕，则使用 flower-small.jpg。而对于小于 600 像素宽的高密度屏幕，则使用 flower-small-HD.jpg。

如果设备的条件发生变化，图像能够被立即替换。例如，如果你旋转平板电脑屏幕使得视口宽度发生变化，浏览器会检查它是否需要从 srcset 下载另一不同的图像。

注意，你可以通过最大视口宽度指定图像（如本例中 600w），它具有与 max-width 媒体查询相同的效果（如果视口最大是 600 像素宽，则使用这个图像）。但遗憾的是，srcset 只能用于最大宽度，不能用于最小宽度。

不支持 srcset 的老式浏览器将忽略其他选项，并使用 src 属性中的默认图像。

2. **<picture>**

与 srcset 一样，<picture> 元素允许你指定并上传同一图像的多个版本，浏览器只会下载它需要的那个文件，而不是所有的文件：

```
<picture>
<source media="(min-width: 45em)" src="images/flowerlarge.jpg">
<source media="(min-width: 18em)" src="images/flowermedium.jpg">
<source src="flower-small.jpg">
<img src="images/flower-small.jpg" alt="a flower">
</picture>
```

你可以在这个例子中看到，media 属性使用的语法与 CSS 媒体查询非常相似。你可以用 min-width 或 max-width 来查询。

对于大于等于 45 em 的视口，浏览器将使用第一个 <source>：flower-large.jpg。对于大于等于 18 em，但小于 45 em 的视口，浏览器将用第二个 <source>：flower-medium.jpg。而对于其他视口（小于 18 em），浏览器将使用第三个 <source>（flower-small.jpg），该条语句中不含媒体查询。

你会注意到默认图像放在最后，这正好与我们在 CSS 中使用媒体查询的顺序相反，在 CSS 中我们是先以默认值开始，后面放置媒体查询。

不支持的 <picture> 元素的浏览器将忽略它，并使用备用的 <img> 元素，其位于 <picture> 元素之内。

如果你既要基于视口宽度的媒体查询，又要使用具有不同分辨率版本的图像，你可以组合使用 <picture> 元素和 srcset 属性：

```
<picture>
<source media="(min-width: 45em)" srcset="flower-large.jpg 1x,
flower-large-hd.jpg 2x">
<source media="(min-width: 18em)" srcset="flower-med.jpg1x,
flower-med-hd.jpg 2x">
<source srcset="flower-small.jpg 1x, flower-small-hd.jpg2x">
<img src="flower-small.jpg" alt="a flower">
</picture>
```

6.7.2 其他解决方案

还有其他响应式图像解决方案可用，但它们主要是腻子脚本。腻子脚本是一段代码，复

制新的 HTML/ CSS 功能至旧浏览器中。其中的一些复制了拟议中的 <picture> 元素和 srcset 属性的行为。使用腻子脚本的缺点是，它需要向网站增加额外的代码。

没有一种解决方案能完美应对所有的使用场合，所以你需要了解每个方案的优缺点，以及应该在何时使用它们。如果想在响应式图像解决方案之中做出选择，可以参考 CSS-Tricks 上由 Chris Coyier 撰写的“Which responsive images solution should you use？”（http://css-tricks.com/which-responsive-images-solution-shouldyou-use/）一文，文中对多种解决方案进行了比较。

1. Picturefill

尽管实际上你还不能使用 <picture> 元素（因为它还未被浏览器实现），不过 Scott Jehl 创建了一个名为 Picturefill（https://github.com/scottjehl/picturefill）的腻子脚本，本质上是用 JavaScript 做同样的事情。

要使用 Picturefill，访问其官方网站并按照说明进行操作。你需要将 picturefill.js 文件添加到网站中。

Picturefill 代码使用与 <picture> 元素类似的语法：

```
<span data-picture data-alt="a flower">
<span data-src="images/flower-small.jpg"></span>
<span data-src="images/flower-medium.jpg"
data-media="(min-width: 18em)"></span>
<span data-src="images/flower-large.jpg"
data-media="(min-width: 45em)"></span>
<noscript><img src="images/flower-small.jpg" alt="a
flower"></noscript>
</span>
```

Picturefill 的 JavaScript 脚本使用 <span> 元素为视口创建一个合适尺寸的元素。在 <noscript> 元素中包含的图像则是为不支持 JavaScript 的浏览器准备的。

对于高密度屏幕，你也可以使用 Picturefill 基于分辨率为浏览器赋予图像选项。

2. Adaptive Images

Adaptive Images（自适应图像）是另一种用于响应式图像的腻子脚本。它是一个服务器端解决方案，这意味着网站托管服务器需要做一些准备工作，而不仅仅是 HTML 和 CSS。

其工作原理不同于先前描述的响应式图像解决方案，你只需要创建一个全尺寸版本的图像。Adaptive Images 检测用户的屏幕尺寸（不是视口），并为之创建和提供一个与屏幕相称的经过尺寸调整的图像版本。

要使用此腻子脚本，服务器需要运行有 Apache 和 PHP，所以 Adaptive Images 并不适用于所有网站。但只要服务器上有这些（它们很常见），你可以很容易为任何网站加入 Adaptive

Images 功能，因为它完全不需要为图像改变 HTML 代码，你只需使用普通的 <img> 元素。

因为不用改变 HTML，这很可能是添加响应式图像到现已有网站的最好方案，这些网站不可能退回去改变原有遗留内容的代码。

要在网站上安装 Adaptive Images，从 Adaptive Images 网站下载文件（http://adaptive-images.com），然后按照说明安装，包括编辑网站的 .htaccess 文件，并添加一些 JavaScript 到页面的 <head> 元素中。

然后在你希望改变图像尺寸的位置（视口宽度）选择并设置断点。因为上传的是图像的最大尺寸版本，断点是在此宽度上该图像将变小。（请注意，断点是以像素单位设置的，而不是我们在本书前面的例子中使用的 em 单位。）

所以，如果你的断点设置为 800 像素和 400 像素，并且上传的图像是 1000 像素宽，其将被调整成两个尺寸的版本：800 像素宽和 400 像素宽（当然，图像的高度也会作相应的调整）。

浏览器加载网页时，每当它遇到一个 <img> 元素，就从服务器请求图像。Adaptive Images 代码将检查用户屏幕宽度，并挑选大于等于屏幕宽度的最小的图像。因此，如果屏幕宽 320 像素，其将选取 400 像素的图像（基于在先前设置的断点）；如果屏幕宽度为 700 像素，它会选择 800 像素的图像。

如果你上传的图像只有 500 像素宽，这是你需要它在屏幕上的最大宽度，那么它只会被调整为 400 像素的断点。图像决不会变得更大。

当一个图像被请求时，服务器会根据请求的尺寸自动创建一个文件。然后，它会缓存图像以节省下一次请求的图像创建开销，因此，仅有在断点范围内第一个访问网站的用户会有轻微的延迟，因为会创建额外的图像。

腻子脚本的一个缺点是它仅根据屏幕尺寸来调整图像的尺寸，而不是图像将要被显示的尺寸。所以不按全屏宽度显示的图像不太可能获得一个按比例缩小的图像版本，即使存在这样合适的版本。

此腻子脚本需要 JavaScript 才能正常运行。在不支持 JavaScript 的浏览器中，Adaptive Images 将不会运行，浏览器会显示 <img> 元素中原始的全尺寸图像。

Adaptive Images 是根据知识共享协议（Creative Commons）授权的，只要保留作者 Matt Wilcox 的署名即可免费使用，此外也有适用于 WordPress 和 Drupal 的插件可供使用。

3. HiSRC

HiSRC（https://github.com/teleject/hisrc）是一个由 Chris Schmitt 编写的 jQuery 插件，允许基于网速和屏幕分辨率来切换图像。

对于每个图像，浏览器会首先加载一个低分辨率的“移动优先”版。浏览器通过 JavaScript 检查连接的速度，并且如果设备使用的是如 3G 这样的移动通讯网络，图像会保持为低分辨率版本。如果有更多的可用带宽，它会下载一个更高分辨率的版本，并用新的图像文件替换原来的低分辨率图像。如果还检测到屏幕是高密度屏，它也将下载合适的版本并替换原图像。

要使用 HiSRC，你需要添加 jQuery 到网站中，并上传和链接 HiSRC 的 JavaScript 文件。你还需要为每个图像制作并上传了三个版本的副本。

至于 HTML 则很简单，你需要一个具有恰当 class 属性的 <div> 元素，且 <img> 元素需要额外的属性来指定三个图像文件：

```
<div class="hisrc">
<img src="images/flower-mobile.png" data-1x="images/
flower-400x200.png" data-2x="images/flower-800x400.png">
</div>
```

[小贴士]

你不必对网站上的所有图像都使用 HiSRC。如果你不想将其用于特定的图像，只需使用 <img> 元素而不要额外的 <div> 元素。另一个提供响应式图像的 JavaScript 插件是 Foresight.js（https://github.com/adamdbradley/foresight.js）。

4. 第三方服务

有数家公司提供了能自动调整图像尺寸以适应屏幕宽度的响应式图像服务。比如，我们可以看下 Sencha.io SRC（http://www.sencha.com/learn/how-to-use-srcsencha-io/）。这项第三方服务对小型网站是免费的。

[小贴士]

其他提供响应式图像服务的第三方网站还有 ReSRC（http://www.resrc.it）、Thumbr.io（http://www.thumbr.io）和 Responsive.io（https://responsive.io）。服务价格通常基于图像的数量或每月的总流量。

第三方响应式图像服务的工作方式是首先检测设备的品牌和型号，然后利用这些信息来确定屏幕的大小。它缩放图像至设备屏幕宽度，并在高速缓存中保留图像 30 分钟，因此相同设备发起的后续请求会变得更快。

所有你需要做的就是添加 Sencha 服务器的 URL（http://src.sencha.io/）到图像的 src 属性中。

例如，如果这是你的 <img> 元素：

```
<img src="http://www.example.com/images/butterfly.jpg" alt="butterfly">
```

你可以像下面这样加入 Sencha 服务器的 URL（请注意，也必须对图像使用完整的 URL，比如 http://www.example.com/images/butterfly.jpg，而不是像这样的相对链接）：

```
<img src="http://src.sencha.io/http://www.example.com/images/
butterfly.jpg" alt="butterfly">
```

图像将被调整到设备屏幕宽度的尺寸。这意味着如果图像实际是以百分比宽度显示在屏幕上，该图像文件将大于它理想的宽度。

如果想要调整图像为特定的尺寸，则可以在 URL 中包含宽度和高度，比如像下面的例子那样。或者只指定宽度（会自动限制高度）：

```
<img src="http://src.sencha.io/400/200/http://www.example.com/
images/butterfly.jpg" alt="butterfly">
```

更有用的是，你可以使用百分比。如果你想要调整一个图像的尺寸，并且知道在设计中它决不会占据超过 50% 的屏幕宽度，下面的代码将调整其大小使其决不超过设备屏幕宽度的 50%：

```
<img src="http://src.sencha.io/x50/http://www.example.com/images/
butterfly.jpg" alt="butterfly">
```

甚至你可以指定 50% 的宽度，25% 的高度：

```
<img src="http://src.sencha.io/x50/x25/http://www.example.com/
images/butterfly.jpg" alt="butterfly">
```

Sencha 还可以做一大堆其他的事情，包括缩减图像尺寸至给出的像素量，使用公式（上述参数的任意组合）处理图像，以及改变文件格式。

请记住，在使用第三方服务时，网站部分内容的呈现会依赖于此服务。而像 Sencha 这样的第三方服务总会有极小的可能性在毫无征兆的情况下突然不复存在，这将从根本上弄垮你的网站。更常见的可能则是有时第三方网站暂时关闭或变慢，这都将影响你的网站的性能。

如果你不想依赖第三方解决方案，并且你有一个值得付出努力的非常大的站点，可以在你自己的服务器上建立一套类似的系统。

6.7.3 断点

这其中的一些响应式图像解决方案需要选择一个或多个断点（也就是视口宽度，在这个宽度上图像对应的文件将发生变化）。那么该如何来选择这些断点呢？

如果响应式图像是以不同视角的剪裁版来显示图像，而不仅仅是同一图像的不同分辨率，你可能需要使图像断点与布局断点保持一致。

而如果是在不同的分辨率下使用同一图像，则根本无需依赖布局。取而代之，你应该是在要下载的文件并不需要那么大时，将图像切换到另一个版本。

如果你使用 Picturefill（我们会在下一个例子中用到）或者其他的解决方案，它们允许你为每个图像单独设置断点，因此你需要做一点额外的工作来使每个图像都是最好的选择。

首先确定该图像在屏幕上的最小和最大尺寸，这样你就知道图像的宽度限制是多少。接着你需要确定两个图像各自的文件大小，接着来决定它们之间的断点设置。

各断点之间相距多远是随意的。如果网站的每个页面上都有大量的图像，你可能希望断点设置得更密一些，从而耗费最少的带宽。当然，这意味着要创建更多的图像文件，对应每种尺寸变化都需要创建一个图像版本。

或者，如果网站的每个页面上只有少量图像，你可能认为只为节省几 KB 根本不值得做这个额外的工作。

以 20 KB 的文件大小作为图像之间的间隔是一个好的开始。尝试调整图像为不同的尺寸，检查文件大小，并根据文件大小每隔 20 KB 设置一个断点。

例如，如果显示在设计中的图像宽度是在 320 像素和 960 像素之间，你最终可能会得到像表 6-1 所示的图像列表。

表6-1：可能的图像断点示例

图像大小	文件大小
960 × 720	108 KB
780 × 585	85 KB
600 × 450	63 KB
500 × 375	43 KB
320 × 240	19 KB

为区分不同版本图像，你可以在文件名中包含宽度，比如 panda960.jpg。

接下来，你需要对这些文件逐一设置媒体查询。如果图像是类似横幅广告总是以全宽视口显示，那简单，只需将断点与图像宽度相匹配即可。

在开始时应该将最大的图像作为默认图像。对于不支持 JavaScript 的浏览器，它只会显示默认的图像。你把最大图像作为默认图像，那么它将适用于所有的屏幕尺寸。

因此，对于最宽的屏幕，我们显示图像 panda960.jpg，其宽度为 960 像素（我们已经确定这将是被显示图像中最宽的），它既可以以 960 像素显示，也可以按比例缩小至较小的尺寸。

对于第一条媒体查询，它的意思是说如果视口宽度小于等于（最大不超过）780 像素，浏览器应该使用 panda780.jpg 代替默认图像。图像可能是以 780 像素的宽度显示，也可以缩放到更小的尺寸，但绝不会大于其实际尺寸。

注意，在这个例子中，我们使用的度量单位是像素而不是 em，因为我们是以像素为单位匹配图像尺寸的：

```
<span data-picture data-alt="A panda eating bamboo.">
    <span data-src="images/panda960.jpg"></span>
    <span data-src="images/panda780.jpg"
    data-media="(max-width: 780px)"></span>
    <span data-src="images/panda600.jpg"
    data-media="(max-width: 600px)"></span>
    <span data-src="images/panda500.jpg"
    data-media="(max-width: 500px)"></span>
    <span data-src="images/panda320.jpg"
    data-media="(max-width: 320px)"></span>
    <noscript><img src="images/panda320.jpg" alt="a
    flower"></noscript>
</span>
```

请记住，以上我们创建了该图像大小依次相差 20 KB 左右的 5 个不同版本的文件，但这只适用于这个特定的图像。如果是对另一幅图像做同样的事情，可能要创建的版本数量不同，宽度也会有所不同。因此，以这种方式优化图像，能将带宽的浪费减至最小程度，但无疑也会涉及大量的手工工作。

另一种选择是只选几个断点并根据这些断点调整所有图像的尺寸。你仍要为每个图像创建各种尺寸的副本，但你不必花费时间计算文件大小来确定每个图像的最佳断点。当然缺点是无法像逐一计算每个图像的文件大小那样能节省那么多的带宽。

要想获得图像断点选择的更多信息，阅读 Jason Grigsby 在 Cloud Four Blog 上的文章“Sensible jumps in responsible image file sizes”（http://blog.cloudfour.com/sensible-jumps-in-responsive-image-file-sizes/）。

6.8 总结

不管是对于网站的内容还是对于网站的设计布局，图像都是一个关键部分。

我们可以在网站上使用 JPEG、GIF、PNG 或 SVG 格式的图像文件。应该使用哪种文件格式取决于你拥有哪种格式的图像。应使每个图像文件尽可能的小，以减少下载它所需的时间。

在网站上显示图像时，首先确定你是否真的需要用到图像，还是可以用 CSS 样式来代替。内容图像使用 `<img>` 元素，装饰图像使用 CSS `background-image` 属性。拼合图像和图

标字体可以帮助你降低图像所需的带宽。

在响应式网站上图像应具有灵活性。使用 `max-width` 来确保图像不会超出包含它们的元素。你可以在布局中浮动图像来获得文字环绕效果。

内容图像需要设置替代文本，这样在无法看到图像的时候，它们的信息仍然可被看到。确保你的替代文本提供与图像相同的信息，而图像信息又应与其使用场景相关。

高密度屏幕需要更高分辨率的图像，但应该避免让那些并不需要高分辨率图像的设备下载它们。有很多种方法可用于让浏览器只下载它需要的图像。

`<picture>` 元素允许你根据视口的宽度指定不同的图像，但它尚未被浏览器实现。现在可以用到的一个选择是 Picturefill 腻子脚本，其他选择包括 Adaptive Images 和诸如 Sencha.io 这样的第三方服务。

选择图像切换的断点时，应基于要下载图像的尺寸，当不再需要那么大尺寸的图像时应切换成更小的图像文件。

响应式图像领域仍在不断发展之中，请务必保持对该领域出现的新思路和新解决方案的关注，以便我们能第一时间用上。

现在，我们已经接触了创建响应式网站涉及的所有主要知识。在第 7 章中，我们将开始学习创建响应式网站的流程。

第三部分

玩转响应式设计

第7章 响应式设计工作流程

对响应式网站有了全面的了解并不等于就能够创建响应式网站了，它们完全是两码事。

在本章中，我们将审视创建响应式设计的全过程，首先从用户研究和内容策略入手，然后是以文本模式进行设计，绘制草图及创建响应式原型。

我们将学习样式板及其他新的设计方法来取代在创建固定宽度的非响应式设计中所使用的Photoshop之类的传统工具。

在本章结尾，我们将讲述如何向客户与同事推销响应式设计，以及在进行响应式项目设计时如何调整工作方式来与客户共事。

如果想要更加深入地研究如何调整工作流程来制作响应式网站，可参考Stephen Hay所著*Responsive Design Workflow*一书（http://responsivedesignworkflow.com）。

7.1 策略与规划

在开始考虑如何设计网站之前，你需要先退一步思考下网站的目标。

除非你的目标并不是创建一个真正的网站，而只是为了好玩或练下手，否则目标首先应是解决如何与客户沟通以及如何在线销售产品等问题。

网站是你创建出来解决以上问题的工具。

你应该从一开始就明了网站或项目的目标是什么，不管这个目标是通过你自己与客户或利益相关者交流得来的，还是从项目经理或其他员工传递来的信息中获得的。“我们公司需

要一个网站”这并不是一个目标。我们应该更深入地挖掘，去发现哪些群体是企业试图沟通的对象，以及哪些目标是其希望网站能达成。

创建一个网站，不管它是不是响应式的，都不是一个简单的过程。在创建过程中要做出很多的创意决策，从网站要包含的内容，到这些内容的摆放位置，以及用户进入网站的路径，等等。如果你明了网站的目标所在，那么在设计过程中你将做出更好的决策。

[小贴士]

我在本节中大量使用了“客户”这个术语，但它并不是单单指雇用设计公司创建网站的客户。

只要你是为他人制作网站，那个人或团体就是你的客户。因此，如果你是公司内部负责网站的员工，那么项目的利益相关者也将被视为你的客户。

7.1.1　用户研究

在项目开始的时候，你很可能会进行用户研究并创建人物角色、情景以及其他在网站设计过程中起指导作用的资源。

在很大程度上，这些工作并非是响应式设计所特有的，所以在这里我们不逐一细说。如果你想对此有更深入的了解，有很多非常好的资源，特别是 Dan Brown 的著作《高效设计沟通之道（原书第 2 版）》（*Communicating Design: Developing Web Site Documentation for Design and Planning*，*Second Edition*，http://communicatingdesign.com），它将告诉你如何在网页设计过程中创建所需的文档，比如人物角色、流程图、线框图、竞争性分析和可用性报告。

不过，一个特别针对响应式设计的警告是：在创建人物角色时，应避免将其与设备类型相对应。你不是为“移动用户”或“桌面用户”创建网站，你是为每个可能在不同时间使用不同设备访问网站的用户创建网站。

[小贴士]

如果你从未做过用户研究，对人物角色或情景这些术语一无所知，查看 Usability.gov 上的“What & Why of Usability”（ http://www.usability.gov/what-and-why/index.html ）一文，尤其是用户研究基础及术语表部分。

7.1.2　内容

无论什么网站，内容都是最重要的部分。在设计过程中应将内容放在首要位置，而不是到

最后才设法将其塞入到已完成的视觉设计中。

在第 2 章中，我们讨论了如何为网站制定内容策略以及在启动设计流程时首先进行内容审计。我们也谈到了如何创建能在所有尺寸的屏幕上都有良好显示效果的内容。

1. 信息架构

一旦你明确了网站需要的内容，下一步就是对内容进行组织和标注，以及构架网站结构。这部分过程称作信息架构（IA，Information Architecture）。

以往网站所采用的通用性 IA 原则仍适用于响应式网站，但你需要确保网站的架构在小屏幕上也是适用的，因为它们可能没有足够的空间来容纳大的或复杂的导航。

还应牢记，网站架构不是一成不变的，IA 及设计应足够灵活以能适应未来组织架构或项目变化的需要。如果没有为此留出余地，在网站上线后，你终究会感到添加任何新的内容都像是往现有的箱子中硬塞进原本不在其中的东西一样。

> **[小贴士]**
>
> 不熟悉信息架构理念？请查看 Martin Belam 发表于 *The Guardian* 上的 “What is ‘Information Architecture’?” 一文（http://www.guardian.co.uk/help/insideguardian/2010/feb/02/what-is-information-architecture）。

2. 内容大纲

在开始组织需要出现在新网站上的内容时，首先应该创建一个简化的内容区域高级大纲，其将反映在网站的主导航中。

例如，图 7-1 是 Mule Design 公司网站的主导航截图。截图下面是一个简单的内容大纲。

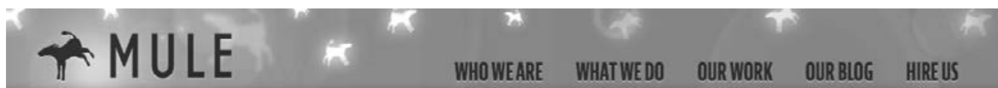


图 7-1：Mule Design 公司网站上的主导航简单明了

- Home（首页）
- About (Who We Are)（关于我们）
  - Individual Bio #1（个人简介 1）
  - Individual Bio #2（个人简介 2）
  - Etc.（其他）
- Services (What We Do)（服务）
- Portfolio (Our Work)（作品集）

    - Client #1（客户 1）
    - Client #2（客户 2）
    - Etc.（其他）

- Blog (Our Blog)（博客）
    - Blog Entry #1（博客文章 1）
    - Blog Entry #2（博客文章 2）
    - Etc.（其他）

- Contact (Hire Us)（联系我们）

你会发现在许多设计公司网站上，顶层的内容区域基本上都有相同的项：Home（首页）、About（关于）、Services（服务）、Portfolio（作品集）、Blog（博客）和 Contact（联系信息）。也会注意到网站上的文字与大纲中的不同："Who We Are"（我们是谁）取代了"About"（关于）。

当你是首次创建内容大纲时，最好是使用诸如"关于"这样的能让你清楚知道此处应放入什么内容的通用术语，而不要去费神考虑页面标题或导航的特定措词。否则你会过于关注诸如术语在导航中是否合适，是否有近似的长度等问题。

当你进一步编辑或开发网站内容时，你可以对为每一部分添加更多的细节。

如同内容清单一样，你不需要列出每个单项（比如所有博客文章或者电商网站上的所有商品），只需把类别和主要内容项写下来即可。

7.2 内容先于布局

在完成了内容策略的制定和信息架构的创建之后，你可以开始考虑内容如何在屏幕上编排。

内容是一个网站最重要的部分，所以你必须围绕内容来创建设计，而不是以相反的方式。

最好的办法是一开始创建一个包含所有内容的无样式网页。这个页面包括的所有页面组件都应以语义化 HTML 代码（比如适当标题级别）标记。

这项工作应在开始绘制草图、线框图或原型之前进行，使得你能在操心内容的显示效果之前先思考每块内容在页面上所扮演的角色。

7.2.1 内容组件

首先，我们要决定哪些内容组件会出现在页面上，比如网站 Logo、搜索框、主导航栏、正文内容和页脚。

在本阶段的设计过程中你需要把这些组件作为单独的内容块考虑，这样在创建原型时你就可以在页面上随意移动它们的位置。

例如，普遍认为网页的“页眉”作为一个固定单元，应包含网站 Logo、搜索框、导航等内容项。但是你需要跳出这种思维框框，在设计响应式网站时并不是所有这些项目都必需放在页眉区域中。

我们可以包含的站点组件如下：

- Logo
- 搜索框
- 主导航
- 次导航
- 信息链接
- 版权声明
- 广告
- 社交媒体链接
- 登录链接

根据页面的类型，可包含的页面组件如下：

- 标题
- 主体内容
- 次要内容
- 简介
- 作者
- 日期
- 引文

7.2.2 以文本模式进行设计

接下来，把所有需要的内容组件放到页面上并用基本的结构性 HTML 元素标记它们（我们在第 3 章讲过如何做到这一点）。例如，在图 7-2 和图 7-3 中，你可以看到那一章的示例网页在添加 CSS 之前的显示效果。

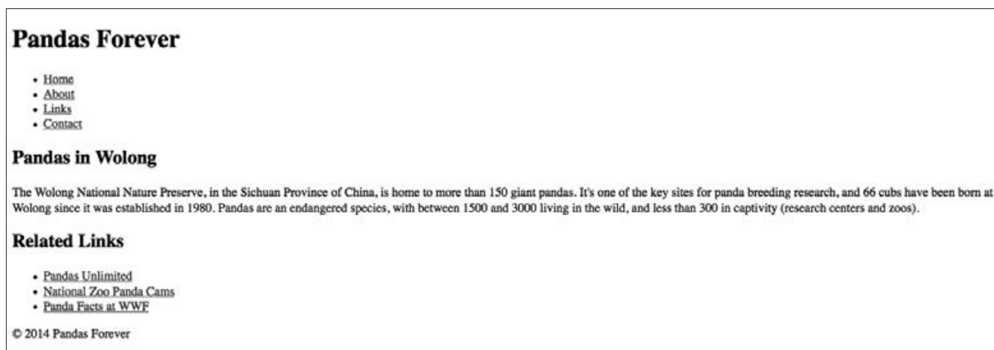


图 7-2：我们的网站未应用 CSS 时在桌面显示器上的显示效果

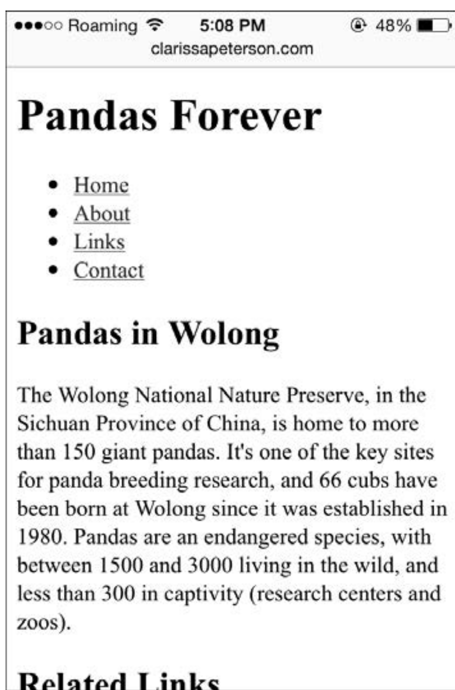


图 7-3：我们的网站未应用 CSS 时在 iPhone 上的显示效果

记住，对一个响应式网站，首要的设计是没有设计，也就是说，你首先创建的应该是用户在其浏览器或设备不支持 CSS 和 JavaScript 时所看到的设计。

仅以此非常基本的代码，你就已经拥有了一个兼容手机（mobile-ready）、响应式和可访问的功能网页。通过这种方式进行设计，你可以确保网页对所有用户都是可用的。

从一开始就使页面对于每个人都是可访问的要比到后期才尝试为此添加可访问性和兼容性要容易得多。

7.2.3 线性设计

当你在页面中加入结构化文本时，你其实就是在创建所谓的线性设计。

线性设计是一个非常简单的想法。想想人们从头到尾阅读网页上的所有内容，你想要他们按什么顺序来阅读，你的 HTML 代码就应该按什么顺序编写。

这个要求看起来再基本不过，但传统上通常是先做视觉设计，然后编写代码以适配设计，再将内容塞入空白区域。这常常会导致实际的 HTML 代码的顺序很奇怪。

而对于小型移动设备，所有内容基本上都是以单列形式显示，因此用户总是以线性顺序来阅读内容的。

开始时就将线性设计运用于无样式文本，这样你就使其一开始就能在小屏幕上工作，同时也能在无法显示你的样式的浏览器中工作。

7.2.4 内容层次

在这个时候你需要做出某些决定，哪些内容部分是最重要的？

最重要的内容应该首先出现在页面上。想象某人从上至下阅读你的页面。在页面顶部，你很可能会有网站标题和（或）网站 Logo，这样当用户到达页面时，就能知道他们所在的位置。页面标题也应放在靠近顶部的位置，以使用户能知道页面是关于什么内容的。

记住此时你并不是在做最终的设计决策，所以不必执着于一切都丝毫不差，你可以在之后随时做出更改。

在这个时候，你也应避免从布局角度去考虑内容，因为在响应式设计中内容不会总是在固定的地方。你可能习惯于把某些元素放在侧边栏中，但某些屏幕尺寸可能没有空间容纳侧边栏，所以这些元素就不得不置于主要内容之上或之下。把你主要的精力放在考虑内容的层次结构上。

7.3 考虑布局

一旦你知道每个页面将包含什么内容，你就可以开始规划每个页面将是什么样子。你需要思考页面设计在不同尺寸屏幕上会是怎样的显示效果。

在转向正式的线框图或原型之前，最好先绘制草图。

7.3.1 草图绘制

在 Stephen Hay 的《响应式 Web 设计全流程解析》（http://responsivedesignworkflow.com）一书中，把草图绘制称为“平面上的思考”。你可以画只有少量细节的小草图，快速尝试

多种想法。之后，当想法更具体化时，可以转而绘制更详细的草图。

草图的绘制从思考网站在各种尺寸的屏幕上（从小手机至大屏显示器）将如何显示开始。

不要担心自己是否会“画”，只需画简单的图形和线条即可，即使线条歪斜也没有关系。

记下什么有用和什么没用，这样你可以把这些想法融入到之后的设计。你的初步草图并不会交付给客户，它们只是一个设计工具。你可以决定是否与团队成员共享最有帮助价值的草图。

想一想各种不同的屏幕尺寸，手机、小平板、大平板，等等，画出与它们大小近似的矩形，然后在里面画你的布局想法。并不需要用尺子去量矩形，你并不是去针对特定的设备设计断点，你只需着眼于常见的设备类别。

既可以在纸上画草图，也可以用触控笔直接在设备上画，而且你会发现这比去想象一个设备屏幕要更方便，如图 7-4 所示。Paper（http://www.fiftythree.com/paper）是一个很好的绘制草图的 iPad 应用。



图 7-4：可以用触控笔直接在设备屏幕上画初步草图（图片来源：Baldiri，http://www.flickr.com/photos/baldiri/5734993652）

7.3.2 从小屏幕开始

不管你是从草图还是从原型开始，最好都是遵循小屏幕优先的原则。设计应先从最小的屏幕尺寸开始，然后直至最宽的屏幕尺寸。我们将在第 8 章中讨论如何确定你应该为之设计

的屏幕尺寸范围，但通常最小的屏幕尺寸会是手机尺寸的屏幕。

为什么不是从大到小呢？毕竟，桌面尺寸的设计将包含更多的细节，按从大到小的方式来处理不是更容易吗？

事实上，恰恰相反。我喜欢用这个比喻来说明此道理：想象你住在一套公寓中，你有机会搬入一个更大的房子，那里会有足够的空间存放你的家具和物品，甚至还能在墙上再挂一些图片或其他饰品。

然后再想象一下从大房子搬到小公寓中，所有东西都堆挤在一起，箱子靠墙堆放着，因为没地方打开它们，即使这样你还不得不丢掉一半的家具。

设想一下类似的场景发生于网站，你会发现试图将桌面尺寸屏幕上的所有内容都搬到小屏幕上也是非常困难的。

而从小屏幕开始要容易入手得多，因为在小屏幕上你最终只会使用确实能融入到设计中的内容，它将迫使你更注重内容，什么内容是真正需要的，而不是因为反正有空间就把一切都随意堆放在那里。

同样的如果你住在一个小公寓里：你没有多余的空间，但你设法以某种方式把需要的一切都放进去。如果你住在一个更大的房子中，有很多的房间，你往往会堆积大量并不真正需要且从未用过的物品。

不过，有时候从桌面尺寸屏幕开始你的设计可能会是一个更好的选择，比如当你对一个现有的网站进行重新设计时，你必须使用现有的固定宽度设计作为响应式设计的一部分。如果你已经有了一个桌面尺寸的设计，且它需要继续保持原样，以原设计为起点可能会更有效。

你也要谨记，首先从小屏幕开始设计并不意味着在设计的过程中就不考虑大屏幕。有时在大小屏幕间来回切换很有帮助，至少在初始阶段是这样的。

7.3.3 移动优先

你大概听过“移动优先”（mobile first）这个词，可能想知道它跟“小屏幕优先”（small-screen first）是不是一个意思，虽然这两个概念有些重叠，但它们并非完全相同。

“移动优先“是由 Luke Wroblewski 在其 *Mobile First* 一书（http://www.abookapart.com/products/mobile-first，中文版参见《移动优先与响应式 Web 设计》）中提出的术语，这是一种创建网站及网页应用程序的方法，所做的设计优先考虑使用移动设备的用户，以及这些用户与网站的交互方式。它要求你首先考虑移动设备的局限性及能力。“优先”这个词在此处意味着在设计考量中移动设备比非移动设备要更重要。

保证你的网站能在触摸屏上良好工作是很重要的，我们会在第 8 章中讨论设备，并解决触

摸目标大小等相关问题。但当你设计一个响应式网站时，对于设备应有最大的灵活性，你旨在设计一个设备无关的网站，能在任何类型的设备上都良好地工作。

在移动和非移动设备之间不再有明显的分界线。你可以买一台具有触摸屏的台式电脑。你可以把显示器和键盘与手机连接，把它当“电脑”用，界线只会越来越模糊。

所以当我在本书中使用“小屏幕优先”这个短语时，我仅是指设计响应式布局和设计网站的过程。我用“优先”这个词并不是说小屏幕更重要，而是说从时间顺序角度来讲，为小屏幕做设计将先于为大屏幕做设计。

所有的屏幕尺寸都是同等重要的，虽然是从最小的屏幕开始，但最终可能会在设计宽屏幕的显示效果上花费更多的时间。

7.4　原型

在对布局有了大致的想法后，就可以开始创建布局了。

7.4.1　线框图与原型

在开始进行视觉设计之前，你需要有一个好的站点布局骨架，并知道它将如何运作以及内容如何在屏幕上放置。

有两种不同的方式来构想网站布局。传统上，网页设计采用的是线框图（wireframe）方式，也就是显示页面布局的静态草图。

但对于响应式设计，你的设计不是固定不变的，会随屏幕宽度而变化。因此，使用响应式原型（responsive prototype）变得越来越普遍，其基本上是内置响应式 HTML 的线框图。可以用它们来查看在各种屏幕上设计是如何随视口宽度变换而变化的。

7.4.2　线框图

传统上，网页设计过程到了这一步应该是创建线框图。

网站线框图是页面所包含的各个组件将如何放置的模型。图 7-5 显示的是一个线框图例子，网页上的事物都是以一般轮廓表示，特意省略掉了视觉细节（其将在后面予以补充）。

对于固定宽度网站，线框图列出了不同页面组件（标题、导航、搜索框、列，等等）的确切位置。

但对于响应式网站，每个组件的位置并不是固定的，布局会随视口宽度而变化。所以，你可能会发现与其创建一个并不能真正代表响应式网站的线框图，创建响应式原型（有时也叫响应式线框图）会更有效。

7.4.3 响应式原型

原型（这个术语也被用于工业设计等领域）是一个模型，不仅能够演示某个事物看上去会是怎样的，也能演示其将如何工作。

原型不一定是以与最终产品相同的方式制作出来的。例如，当一辆新型号的汽车被设计出来后，通常会制作出一个原型并送去参加车展，以便能在决定该车型是否投产之前先获取消费者的反馈。这些汽车原型是在研究室中单独制造出来的，而不是由通常的工厂组装线生产，它们甚至可能无法驱动。

同样的，你的响应式原型不需要用像真实网站那样的编码方式来进行编码，交互元素也并不需要能工作。你可以使用 HTML 或者原型软件（我们一会就将介绍几个）来构建一个原型。

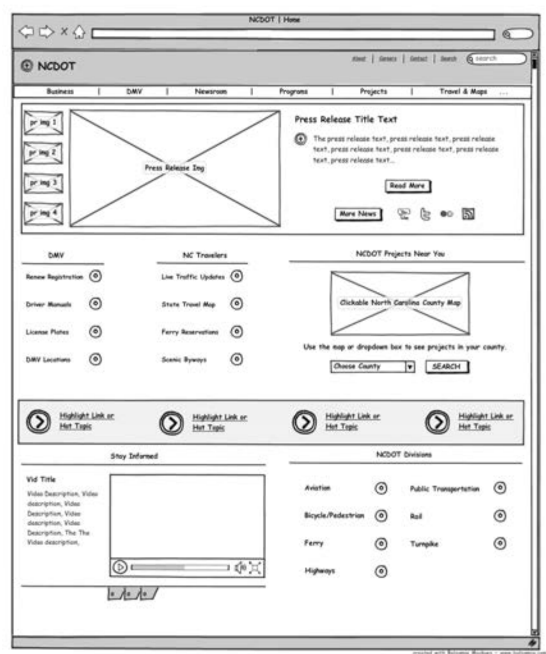


图 7-5：一个固定宽度网站的详细线框图（图片提供者：After Victory，http://www.flickr.com/photos/aftervictory/5097418313/）

响应式原型是一个“真正的”网页，你可以在浏览器中查看它，但它只有基本的布局，显示效果与线框图类似。事实上，它在本质上与线框图是同一事物，区别在于，布局是响应式的，所以你可以调整浏览器窗口或是在不同的设备上查看原型来观察布局是如何变化的。

原型的保真度并无限制：从只有很少的细节到极接近实际的最终产品都行。汽车原型可能看起来像一辆真正的汽车，但也仅用于视觉展示，其缺少引擎和其他的必要元件。或者它可能是一辆能开的车，但只能低速行驶用以演示，缺乏在公路上行驶的安全特性。

同样的事情也适用于网站原型。一般来说，在设计的早期阶段，你将以低保真原型开始，然后随着设计的推进，原型将变得越来越真实。

7.4.4 原型中有什么

原型应该展示布局的基本要素，以及站点布局如何响应视口宽度的变化。

虽然在开始进行视觉设计工作之后，你仍可继续修改布局，但在只关注布局的现阶段，对大量的布局做出合适的决策显然更为容易。

虽然你可能会有几个不同类型页面的原型（如首页、内页），但不要试图链接它们，尽管这样你可以通过点击从一个原型跳转到另一个原型，就如同它们是实际的网站页面一样。你现在应该只专注于布局，链接网页只会增加复杂性。在稍后的设计过程中你可能要制作*交互原型*（interactive prototype），其可以被用来测试网站的实际功能（比如结账过程），但现在你的原型应该只是视觉原型而非互动原型。

7.4.5 从基础开始

你创建的第一个响应式原型只是给出了关于页面上内容的放置及各种视口宽度下内容的层次结构的粗略想法。最终，你会转到高保真原型上来。

你不需要在这里考虑诸如字体或颜色等细节，只需考虑布局。选择一种简单的无衬线字体用于所有内容，并用边框和不同的灰阶来区分页面上的各个内容项，这将有助于你专注布局。如果你使响应式原型看上去太逼真，客户可能会把它当成是“真正的”网站。

如果在制作响应式原型时，从一开始就在各种不同的设备上测试它们，那么所有问题都能在早期得到处理。

当以低保真度原型开始时，你既可以使用占位符代替内容，也可以使用实际内容。使用实际内容的好处是你可以知道它在布局中是否合适。例如，如果在页面顶部放置标题的地方以“页面标题示例”做为占位符，虽然它看起来是完美匹配布局的，但当你后来必须插入一个实际有 10 个字而不是 6 个字的页面标题时，就可能变得不再适合了。

同理，当创建原型时，不要只考虑理想化的页面会是怎样的显示效果。你应该确保你的原型可以应对极端情况——网站中最复杂的页面。

7.4.6 创建页面布局

当开始考虑网站的实际页面时，你必须决定哪些页面或页面类别需要创建布局。

由于设计中的某些部分在大多数页面或者是所有页面上（比如页眉和导航很可能在所有页面上都是一样的) 看起来都是一样的，你只需要为几种类型的页面创建布局。

梳理你的内容大纲并基于内容得出你都有哪些页面类型。

例如，对于一个报纸网站，可能会有的页面类型包括：

- 主页，其是独一无二的；
- 文章页，每个页面包含一篇文章；
- 类目页（比如“本地新闻”类目页包含指向该类目下文章的链接）；
- 图库页，类似于文章页的独立页面，包含的主要是图片而不是文本；
- 信息页，比如“隐私保护”。

你可以在图 7-6 中的华盛顿邮报网站看到以上这些种类的页面布局。

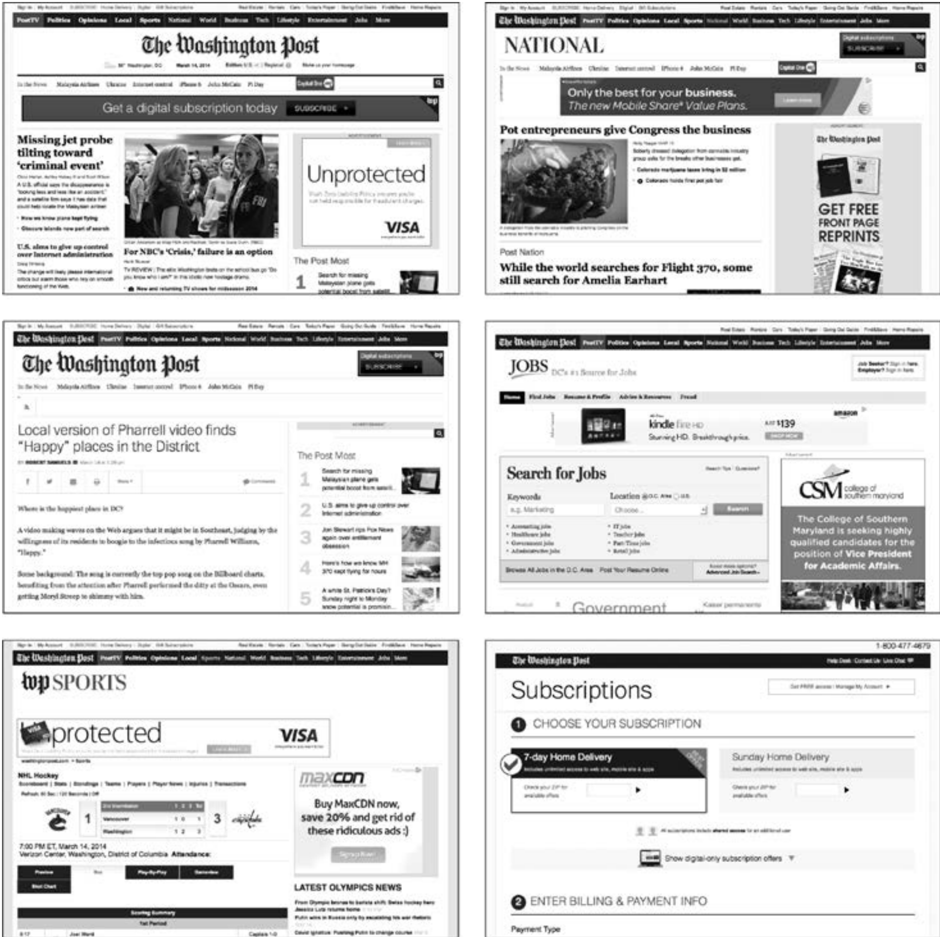


图 7-6：用在《华盛顿邮报》网站上的 6 种页面布局

有些元素在所有的页面上都将始终是一致的。例如，网站上的每个页面在其顶部可能都会有相同的页眉、导航栏和搜索框，尽管不是必须这么做。在图 7-6 中，除首页外（其在导

航栏左侧省略了“华盛顿邮报”的 Logo），所有类型的页面上导航栏都是相同的。

思考每个页面可以如何被分解成不同的组件，这些组件是可在页面上被自由移动或复制到其他页面的独立内容块。

有些页面类型与其他类型非常类似。例如，刚刚描述过的文章页面和信息页面就可能会非常相似，因为它们主要包含的都是文本。但是文章页面需要有地方放置作者和日期等元数据，而信息页面则不需要。

在设计过程中你会为每种不同的页面类型制作一个或多个布局。通常，一个网站会有 2 至 5 种页面类型。这些布局应该是最复杂的布局，因为较简单的布局往往可以从它们之中派生出来。例如，你并不需要为信息页面单独创建一个原型，因为它只不过是去除了某些组件后的文章页面。

先做首页是很有吸引力的，因为它就像是网站的“脸面”。但从最简单的内页开始，在内页都有了确定的布局后再转向更复杂的主页往往会更容易。

7.4.7 框架

要加快原型设计的过程，你可以使用框架（framework）。框架是一个可下载的 HTML 和 CSS 包，包含了创建一个基本网站需要的所有元素。通过框架你可以在一两个小时内就创建出一个响应式 HTML 原型，而不必从零开始。框架也可以作为实际网站的基础代码。

对于一个框架，其包含的 CSS 将为 HTML 元素添加基本的样式。因此，从一开始，你的段落、标题、列表、按钮和表单字段都有一个虽然基础，但看起来还不错的外观，你不必再为它们去写 CSS 样式。框架还包含有说明文档指导你如何使用它提供的 CSS 类来为页面元素（包括列和导航）添加响应式布局能力。

即使你只有 HTML 和 CSS 的基本知识，也能够很容易地使用框架快速地创建一个响应式原型（如图 7-7 所示）。当创建一个实际的网站时，使用框架并不是说就不用知道代码是如何工作的了。但在原型设计阶段，其意味着设计团队成员将能够自行创建或使用响应式 HTML 原型，他们无需拥有开发人员那样的专业技巧，自己就能使用框架和修改代码。



图 7-7：这些原型示例都是用 Foundation 响应式框架创建的

由 ZURB 开发的 Foundation（http://foundation.zurb.com）框架，如你在原型示例中所见到的，是一个极佳的制作响应式原型的工具。它包含一个小屏幕优先的具有语义化标记的 12 列弹性网格。Bootstrap（http://getbootstrap.com）是另一种常用的框架。你也可以在网上搜索“响应式框架”（responsive framework），能找到数十种其他选择。它们具有很大的差异，所以你应该逐一测试以找出最适合你需要的框架。

7.4.8 原型创建工具

如果你不想用 HTML 来创建原型，有几个工具（包括桌面应用和在线应用）可以用来创建响应式原型。

不过，在大多数情况下，这些工具并不能创建真正的可以在任意屏幕尺寸下查看的响应式原型。相反，它们只是能让你基于不同的屏幕尺寸创建数个不同的静态线框图。

下面是一些响应式原型设计工具：

- Balsamiq（http://balsamiq.com/）；
- Froont（http://www.froont.com/）；
- HotGloo（http://www.hotgloo.com/responsive-prototype-tool），如图 7-8 所示。

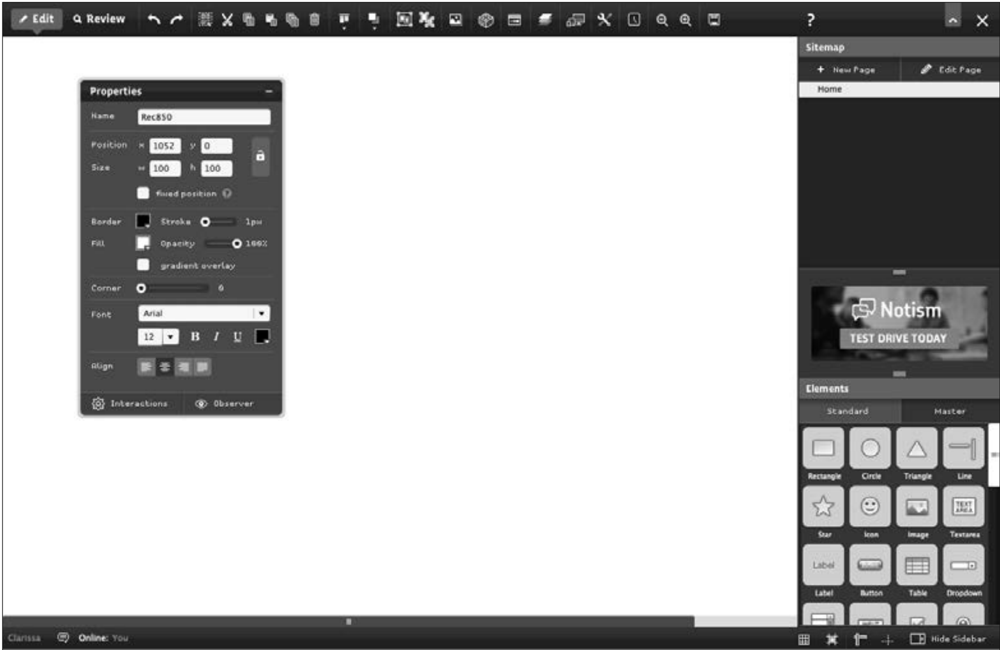


图 7-8：HotGloo 在线原型创建工具

你还可查看 Balsamiq 的文章“Responsive Design with Mockups”（http://support.balsamiq.com/

customer/portal/articles/615901-responsive-design-with-mockups）以获得一些入门的技巧。

7.5 视觉设计

原型创建好后，接下来是进行网站的视觉设计。在这一阶段我们添加颜色、文字排版样式及品牌元素。响应式网站相对固定宽度网站又会有些不同。

传统上，视觉设计是以一个或多个 Photoshop 设计样稿（网站显示效果的平面视觉表现）的形式提供给客户或者利益相关者，来展示网站首页及内页。但是，对于一个响应式网站，你不应仅以一个静态页面视图来展示它们。

最好在设计早期就让客户参与，向他们展示响应式原型。这将有助于确保他们在整个过程中对响应式网站有一个清晰的认识。而当你推进到视觉设计部分时，除了 Photoshop 设计样稿外，还有其他一些方式可用来呈现视觉设计元素。

7.5.1 样式板

如果不打算向客户展示一个像素级精确的设计样稿（响应式网站无需像素级精度），那怎么做才能缩小原型与完全设计好的网站之间的差距呢？我们可以从样式板（style tiles）开始。

设计师 Samantha Warren 想出了样式板（http://styletil.es）这一概念，样式板是由字体、颜色和界面元素组成的能传达网站视觉品牌精髓的可交付设计成果。

Warren 解释说，制作样式板类似于室内设计师设计房间装修方案的过程。室内设计师并不会立即拿出三种不同的房间设计方案给客户，而是首先会与客户一起讨论来决定房间装修设计所使用的颜色、纹理和材料，然后再基于这些选项进行设计。

样式板例子如图 7-9 所示。

使用样式板可以帮助你在交付一个完整设计之前拟订出核心设计元素。

创建样式板从倾听客户需求并向他们提出问题开始，这些你通常在需求收集阶段做的事情。特别是试着去了解用户的想法，他们希望从网站获得怎样的视觉观感。此外，务必明确是否需要遵循现有的品牌或风格指南。

在创建样式板时，你既可以使用像 Photoshop 这样的图像编辑工具，也可直接在浏览器中工作。无论哪种方式，你都可按需要尽可能多地生成不同样式组合以与客户达成共识。现在重做样式要比之后容易得多。当然在后期你也还是可以改变样式的，因为有时在看到样式在实际的网站设计上的显示效果之前，你并不一定能完全明白设计决策所带来的影响。



图 7-9：Examiner 网站（第四幅图）的最终设计基于左边三幅图中的样式板

> [小贴士]
>
> 访问 Style Tiles 网站（http://styletil.es）以更多地了解样式板，你还可下载一个免费的 Photoshop 模板。

7.5.2　测试与调整

响应式网页的设计过程需要随着进度持续不断地测试。只要你有了 HTML 原型，你就应该在不同的设备上测试它以查看显示效果。如果一个布局在理论上似乎不错，但最终无法工作于某个特定的屏幕尺寸，你最好是能在早期发现这样的问题，而不是在网站快要完工时才试图再去调整。

我们将在第 8 章中更深入地探讨如何进行测试。

7.5.3 风格指南

风格指南（style guide）收集和记录对网站所做出的所有设计决策。用哪些字体，用在哪里？正确的字体大小是什么？按钮看起来是什么样子？

根据项目的需要，风格指南可简可繁。它有两个用途。首先，风格指南可以帮助你在开发过程中确保设计的一致性。如果提交按钮应该总是蓝色的，怎么一个表单会出现一个绿色的提交按钮？

矛盾的发生有时是因为我们没有注意到代码中的错误，有时则是因为没有人意识到针对一个特定页面做出的设计决策已经影响到整个网站。如果整个网站具有一致的视觉体验，那无疑会给人更好的感觉。

风格指南的第二个用途是能够帮助网站未来的管理人员做出设计决策，这将有助于维护网站设计的一致性，因为管理人员会随着时间而发生变化。设计网站的人可能都熟知哪种字体用在哪里以及为什么用在那里，但是当客户内部负责网站维护的员工在后期需要添加一个新特性时，如果没有风格指南的指引，他们很难知道应该使用何种字体。

风格指南在印刷设计领域已使用多年，有时也称作风格手册或品牌指导手册。通常，公司将有一个通用的风格手册来说明品牌 Logo 的使用等事项。一个网站的风格指南应以此基础并更加的细化。

除了列出设计元素（字体、颜色等）之外，展示它们的实际效果也是有帮助的，如图 7-10 所示，这样能很容易地一看就知道它们看起来是什么样子。你也可以指定特定的 CSS 来创建特殊的效果。

风格指南并不是只能包含视觉风格；它还可以包含内容风格和编码标准（或其他可放在不同风格指南中的规范）。

尽管在 Word 或其他文字编辑软件中以文档方式来创建风格指南十分普遍，但往往用 HTML 来创建它要容易些，这允许你更准确地展示设计风格在屏幕上的显示效果。否则，你可能要包含示例截图。

下面是你可能希望包含在风格指南中的内容的一些例子：

- 文字排版；
- 颜色和纹理；
- 布局系统 / 网格；
- HTML 元素（诸如列表项、表单元素或块引用）样式；
- 适当的标记（`<h1>` 是用于网站标题还是页面标题呢）；
- Logo 用法；

- 语音；
- 常规风格，比如标点符号与词语用法，如果你不是使用如 *The AP Stylebook*（美联社写作风格）或 *The Chicago Manual of Style*（芝加哥格式手册）这样的商业风格指南 。

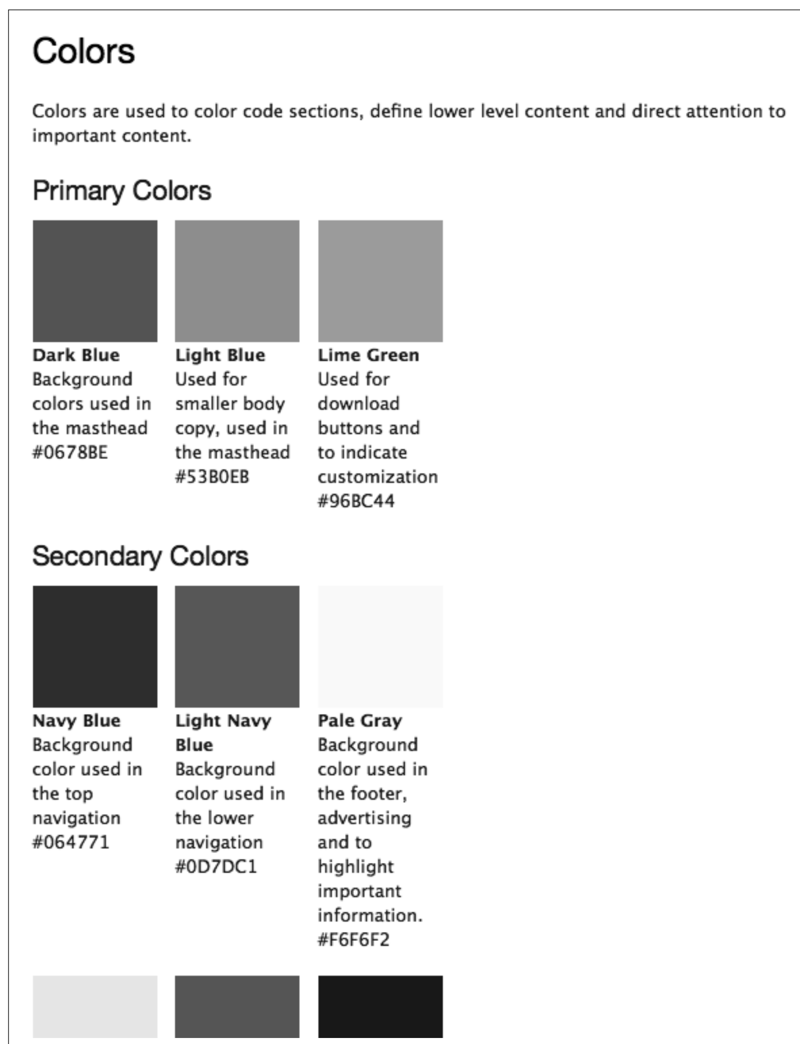


图 7-10：Drupal.org 风格指南中的一页列出了颜色（主要颜色、次要颜色及信息颜色）；每个条目给出颜色名、十六进制值及其用途（比如，深蓝色，#0678BE，用作刊头的背景色）（另见彩插图 7-10）

Brett Jankord 上的 Style Guide Boilerplate（风格指南样板文件，参见 http://brettjankord.com/projects/style-guide-boilerplate/）也十分有帮助。

下面是各种类型的网站风格指南的优秀范例：

- Starbucks.com（http://www.starbucks.com/static/reference/styleguide/，为风格指南）；
- South Tees Hospitals NHS Foundation Trust（http://www.southtees.nhs.uk/style-guide/，为风格指南）；
- Drupal.org（http://drupal.org/coding-standards，为编码标准）；
- BBC 的数字化服务（http://www.bbc.co.uk/gel，为全球体验语言）；
- Paul Robert Lloyd（http://www.paulrobertlloyd.com/about/styleguide/，为 HTML 标记风格指南）。

7.6 响应式设计工具

在设计网站时你应该使用什么工具呢？

记住：正确的工具应能为你所用并能带给你最终的产品。这并没有唯一正确的答案，客户也不关心你使用什么工具，他们只是需要一个网站。

7.6.1 Adobe Photoshop

Photoshop 历来都是设计网站最常用的工具。但是 Photoshop 设计样稿本质上只是一张网站图片。如果你的目标是做一个在所有浏览器中都具有完全相同显示效果的网站（这是我们之前多年来的目标），一张图片确实就能搞定。但是现在，我们试图让一个网站在不同的视口宽度下有不同的外观，一个网站一幅图片显然不再适用。

但在部分设计过程中可能仍要使用 Photoshop，比如设计特定的页面元素或图片素材。

[小贴士]

如果你正在使用 Photoshop，想让它能更好地工作于你的响应式设计过程，可查看 Dan Rose 发表在 Smashing Magazine 上的“Repurposing Photoshop For The Web”（http://www.smashingmagazine.com/2013/04/22/repurposing-photoshop/）。

7.6.2 Adobe InDesign

很多设计师已发现 Adobe InDesign 在响应式设计中更好用。（Photoshop 本来就不是一个网页设计工具，它是用来做图片处理的，而 InDesign 才是用于设计的。从字面上这显而易见，不是吗？）

InDesign 含有网格、样式、包含以及流式页面规则等功能与特性来可以帮助你构建响应式布局。图 7-11 显示了流式布局选项，该选项允许你为多个设备尺寸创建布局。

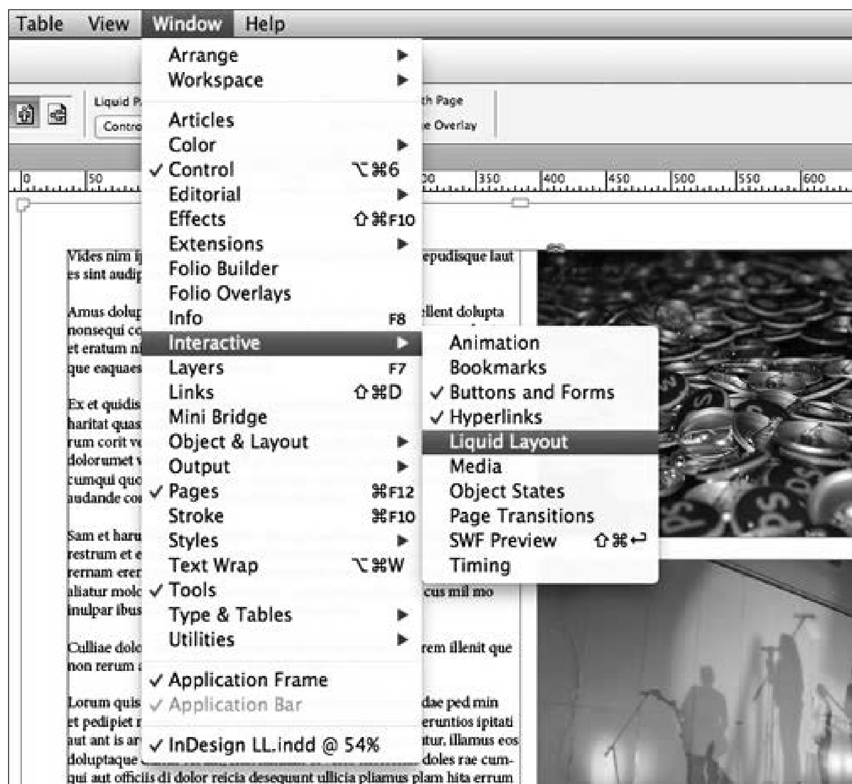


图 7-11：Adobe InDesign 中的流式布局选项

7.6.3 Adobe Edge Reflow

Edge Reflow 是 Adobe 新推出的一种设计工具，允许你创建包含媒体查询的响应式设计，并能输出你设计稿的 HTML 和 CSS 代码交付给开发人员。其用意并不是创建成品级的代码，而是创建可用于不同屏幕尺寸的响应式布局，使设计师能“分享他们的响应式设计意图”。

这有点像在 Photoshop 中创建设计样稿，除了它们是响应式设计以外。

设计师们使用 Edge Reflow 基于网格列来创建响应式布局，并可添加自定义字体或阴影等 CSS 样式。不过 Edge Reflow 的软件界面是纯粹面向设计的，除了输出之外，并无它法可访问实际的 HTML 和 CSS。Edge Reflow 的界面如图 7-12 所示。

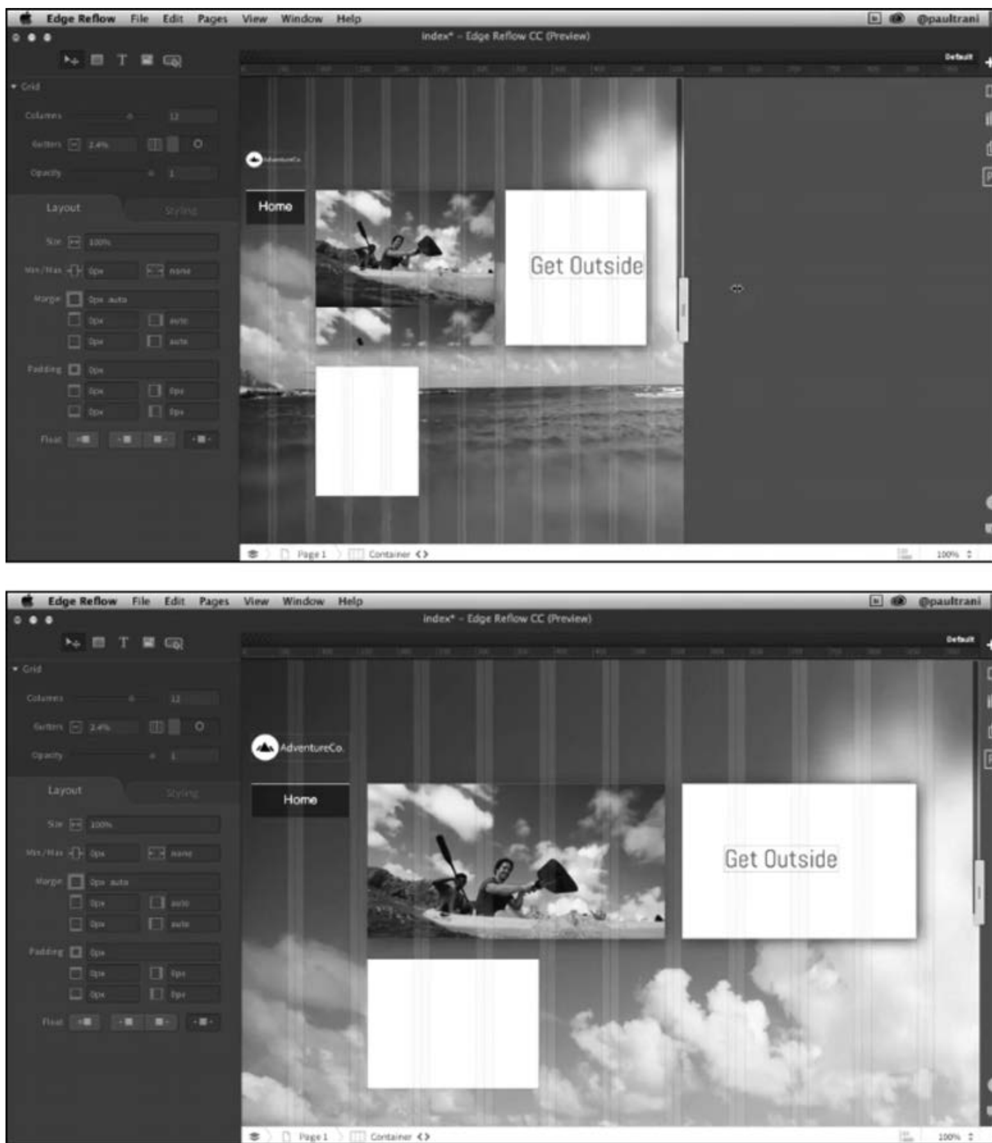


图 7-12：Adobe Edge Reflow（另见彩插图 7-12）

7.6.4　Adobe Dreamweaver

Adobe Dreamweaver 仍被广泛使用，它的最新版添加了一项叫作流体网格布局（fluid grid layout）的功能，允许你创建响应式布局。

遗憾的是，当在 Dreamweaver 中创建一个网站时，你只能指定三种布局，如图 7-13 所示。并不能够自由地为不同的网站组件在它们最合适的断点上添加媒体查询。

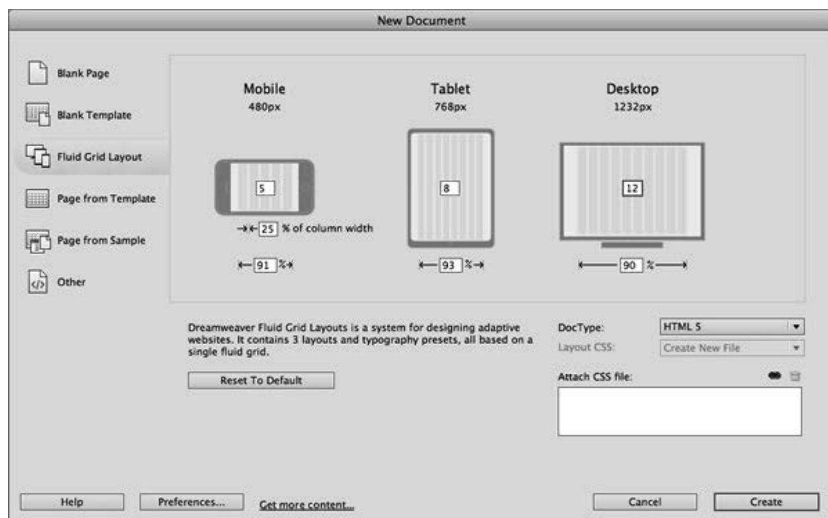


图 7-13：Dreamweaver 的网站布局配置界面，你只能为网站指定三种布局

在每一个布局内部，你可以选择想要的列数，并在网格中添加和移动元素。由此产生的 CSS 使用百分比宽度和浮动样式，因此布局能够适应不同的屏幕尺寸。

7.6.5　浏览器

虽然上述工具都很有用，但许多设计师发现跳过图像编辑软件直接在浏览器中进行设计要更方便。

通过在浏览器中开始设计，可以确保不会出现设计最终在浏览器中有不同的显示效果的情况。

我有很多次作为一个客户，与许多不同的设计公司一起工作。我发现实际网站的显示效果总是与 Photoshop 设计稿有很大的出入，即便对于静态网站也是如此，比如浏览器中的字体渲染效果不同，内容没有以期望的方式对齐等。开发人员尝试全盘复制 Photoshop 设计稿，但他们使用的是完全不同的工具，所以这并没有那么简单。那么为什么要投入这么多的努力使某物看起来“完美”，但事实上在最终的产品中又看起来不一样呢？

通过直接在浏览器中进行设计，你可以使设计一开始就是响应式的，因此毫无疑问网站的外观在所有屏幕尺寸上都会符合预期。

设计师生成的代码不需要与最终在实际网站中使用的代码是一样的，所以不用担心它是否完美。使用浏览器进行设计可以节省后面的时间，尤其当代码还是正确结构化的 HTML 时。

当然直接在浏览器中进行设计也有缺点。你缺乏直接操纵对象（比如线条和图形）的控制手段。如果页面上有图形元素，它们必须分开来设计。

不用浏览器进行设计的设计师在设计过程中应与开发人员密切合作，了解其设计将会怎样呈现在不同尺寸的屏幕上。他们至少应该对 HTML 和 CSS 的工作方式，特别是用于响应式设计的代码（比如媒体查询）的工作方式有一个基本的了解。

如果是直接在浏览器中进行设计，你可以使用电脑自带的文本编辑器创建页面，比如 Mac OS 中的 TextEdit 或 Microsoft Windows 中的记事本。或者你可以使用诸如 Sublime Text（http://www.sublimetext.com）这样的代码编辑器，其界面特别适合编写各种风格的代码。

7.7 推销响应式设计

如果你正在阅读本书，你也许已经被响应式设计的理念所迷住，但很有可能你仍需要向别人推销它。

问题的关键在于如何说服你的同事：我们公司应该考虑将响应式设计用于网站的重新设计。或者，如果你在一家设计公司工作，可能还要说服客户：响应式设计对他们的网站来说是最佳的选择。

不管你是什么角色，你都很可能会与从未听说过或未真正理解响应式设计的人共事。

在本节中，你将了解如何向客户、同事和其他利益相关者推销响应式设计理念。

7.7.1 为什么要用响应式设计

几个月前，我与一家小网站制作公司的两名工作人员聊起响应式设计。其中一人是开发人员，他对能够创建可在所有设备上工作的网站的理念感到非常激动。他们公司还从未做过任何响应式设计项目，但他正在试着努力学习。

另一个人是销售人员。我问他，他们是否尝试过向潜在客户推销响应式设计，他说没有，他们打算继续向客户推销单独的桌面和移动网站。

原因在于：销售团队觉得客户会因为他们能获得两个而不是一个网站而愿意支付更多的钱。

也许确实如此，但是除了有那么一点缺乏职业道德外，它也不是一个良好的商业习惯。对于许多网站而言，如果你想要一个网站在未来数年内都是可用的，响应式设计是最佳选择。虽然有些情况下，单独创建一个手机版站点可能会更好，但通常你应首先考虑响应式设计。

不让你的客户知道他们可以选择响应式设计，意味着客户得到的网站很可能在上线之后不久就会令他们感到不满，因为他们意识到网站不能在所有设备之上工作且不够灵活，跟不上互联网形势的变化。

如果你真的希望公司赚钱，你的目标就应该是让客户在最后结清账款以后很长的一段时间

里仍然感到满意。如果你提供的网站能在数年内满足客户的需求，他们将变成你的回头客并会向他人推荐你的公司。

如果你的设计公司拒绝响应式设计的理由是因为它太难了，或者是因为你认为推销单独的手机版网站能赚更多的钱，你就应该重新思考现在销售的网站在将来是否仍会让客户满意，如果能让客户满意，他们将帮助你进一步地拓展业务，反之，你的业绩就不会长久。更重要的是，能够制作设计良好的响应式网站将给你带来竞争优势，因为还不是谁都知道怎么做。

7.7.2 教育你的客户

你有责任教育客户及潜在客户一个好网站的构成要素是什么。

客户会带着想法来找你，但并非所有想法都是好的，不要被客户牵着鼻子走。合理采纳他们的想法并利用你拥有的专业技能和知识把它们变得更好，那才是他们付给你报酬的原因。

客户就经常会说他们想要一个“手机网站”或“手机应用”。他们单纯地认为要使网站能在手机上浏览必须要另外构建一个单独的网站，并不知道其实还有别的选择。

如果客户来找你，并说他们只是想要一个手机网站，你可以告诉他们摆在他们面前的选择有哪些，这其中包括了响应式设计。如果他们不太了解网站是如何工作的，甚至可能都不知道响应式设计的存在，即使知道，可能对其工作方式也有误解。

这是你的机会，让他们知道响应式设计对其可能是一个更好的选择，因为他们可以只保留一个站点，而不用去维护多个。而且这一个网站就能适用于所有的设备，而不只是某些尺寸屏幕的设备。

你的工作是确定对于客户的每个问题什么是最好的解决方案，并将解决方案推销给他们。如果他们从未听说过响应式设计，但你认为他们的网站应该采用响应式设计方案，就需要帮助其理解什么是响应式设计并让其相信响应式设计是解决问题的最好方案。

7.7.3 强调响应性

记住，你的最终目标并不仅仅是制作一个良好的满足商业目标和用户需求的响应式网站，还应该确保客户对其满意。

如果你创建了世界上最好的网站，但客户对此并不满意，因为它不是他们想要的网站，那你显然就有问题了。

你需要确保客户能理解整个设计过程中所出现的事物。在设计过程的每个环节上都不忘提醒他们你是在创建一个响应式网站，以及其意味着什么。在各种不同的设备上向他们展示

设计，使他们能够明白页面布局在不同的屏幕尺寸上会发生怎样的变化。

如果到了设计的最后阶段，你却发现客户对网站在电脑上与在手机上的显示效果不一样而感到困惑（真的，我听说过有这样的事），那就是你没有把响应式设计理念的沟通工作做好。

响应式设计对人们是一个新的理念。不要认为在你初次解释它后人们就能立刻理解它。不断地向他们展示什么是响应式设计，直到他们能够理解为止。

7.7.4 响应式设计并不总是最佳选择

请记住，响应式设计并非万能钥匙。大多数时候它是一个很好的解决方案，但下面列出的一些原因也使其可能并非最佳解决方案：

- 项目团队缺乏足够的经验来很好地实施响应式设计（但获得的经验教训可让你在下一项目中受益）；
- 客户没有足够的资金支付响应式网站的额外费用（但确保让他们知道响应式设计具有的长期收益很可能会帮他们节省更多的费用）；
- 网站用于短期临时性项目，比如即将举行的活动，且你可以只关注特定的设备，不需担心将来的设备兼容性。

7.7.5 费用

你会从客户那听到一个问了无数遍的问题：做一个响应式网站要多少钱？

他们想要一个具体的答案。比固定宽度网站贵 25%，两倍，还是若干美元？

不幸的是，对这个问题没有一个明确的答案。如何创建一个响应式网站是千变万化的，就如同如何创建任何网站一样。如果你就同样的网站项目向多家设计公司咨询价格时，你肯定会发现报价也是各不相同的。

设计一个响应式网站的费用几乎肯定比同等条件下设计一个固定宽度网站的费用要高。因为有更多的工作要做：布局不是固定的大小，你必须使布局能随屏幕尺寸变化，那么这至少都意味着需要额外添加媒体查询。

费用的高低实际上取决于网站的复杂程度。可能几条媒体查询就够了，也可能存在某些设计元素需要额外的关注。

这还取决于客户想要一个怎样的响应式网站。有很多细节可以深入研究，比如根据屏幕宽度略微调整字体大小以充分利用屏幕空间。尽管这可能使得网站更耐看、更好用，但它对于响应式站点不是必需的。

客户需要更多的费用吗？是的。但是他们不会为同样的最终结果花费更多。响应式网站将能在更多的设备上工作，这意味着他们能增加潜在的用户群。响应式站点的存续周期会更长，所以额外的花费其实是对未来的投资。

需要牢记的一件事情是，如果创建网站的团队成员是响应式设计的新手，那么在开发的过程中，必定要花费额外的时间去学习和尝试新事物。你可能需要付出额外的努力来使某事物正确工作，你可能不得不做研究来找出实现某事物的最佳方式。这些都需要额外的工作时间，你需要对此预先计划，而且通常来说这也不是一个可以转嫁给客户的费用成本。

7.8 与客户合作

正如我前面说明的那样，“客户”一词并不仅仅指设计公司的客户，它也包括企业内部负责网站的员工所服务的内部客户及项目中的其他利益相关者。因此，即使你不是在某个设计公司工作或是只是一个自由职业者，本节对你也很有帮助。

你的客户或利益相关者通常会带着预期目标来到谈判桌上。他们需要一个网站，但他们真正需要的是能解决他们的问题。你需要帮助他们找出理想的解决方案是什么，先询问以下问题。

- 他们想在网站上放什么样的内容？
- 内容的目标受众是谁？是否有分析数据来支持此设定？
- 网站更新的频率，以及由谁负责更新?
- 在现有网站中存在的哪些问题他们希望在新网站中能够避免?

你掌握的信息越多，带给客户的最终结果也会越好。

7.8.1 交付物

响应式网站的交付物与以往固定宽度网站的交付物可能最终会有很大的不同，比如线框图和样稿修订。

你仍然有交付物，如内容架构、响应式原型和样式板，但因为你改变了设计进程的思考方式，你也需要改变交付物的思考方式。

一开始，在设计一个网站时你的工作不是去创建一个交付物，而是去设计一个网站。交付物只不过是帮助确保项目步入正轨这一过程的产物而已。

可能你负责过有初始需求的项目——召集与客户的会面确定目标，在设计团队工作了一段时间后，紧接着是开会向客户展示你建议的设计。客户对你在此期间做了哪些工作毫不知情，可供选择的几个设计方案是刚刚“透露的”。他们只能从中选出一个，在后续的过程中被迫接受该设计。

尽管这是戏剧性和令人兴奋的，但这通常并不会使客户得到令他们满意的产品，因为他们被排除在设计过程之外。更好的选择是让客户能在项目中分享他们的观点。

对于一个响应式网站，你不能像过去常做的那样把交付物作为最终产品，因为在这个过程中需要有更多的灵活性。交付物是一个迭代过程的工作文档，它们不是一成不变的。

你肯定不想陷入这种境地：中间阶段的设计文档被客户“批准”（比如把握财权的高层利益相关方），之后，当发现某个不同的事物能工作得更好时，客户却不允许做出更改，因为先前的工作已经被上司“批准”，他们不想回到上司那里说先前的决定是错误的。你要知道，在设计阶段没有什么必须是一成不变的。

要制作出客户真正满意的网站，你需要意识到客户也是设计团队一份子。让他们从一开始就参与到设计过程中来，并共同做出决定。

当然，在整个过程中都允许做出改变并不意味着允许客户更改项目的范围而不增加费用。通过尽早做出高层决策，并逐步夯实它，你应该能够提供让客户满意的最终设计。

并且记住，并非所有的项目文档都是交付物。用工作文档记录团队的工作，这样每个人都清楚项目进行到了什么阶段，以及在此阶段需要做些什么。这些工作文档不需要与客户共享。

[小贴士]

如果在创建项目文档时需要帮助，你可以参考 Dan Brown 的《高效设计沟通之道（原书第 2 版）》（http://communicatingdesign.com），了解如何创建网站地图、流程图、线框图以及人物角色等细节。

7.8.2 陈述报告

如果你是与一个团队一起工作，在与客户会面向他们展示设计进展时，所有参与设计的工作人员（视觉设计师、用户体验设计师等）都应该在场，那样他们就能获得第一手的反馈。你不需要通过可能不理解发生了什么的项目经理来传达修改意见。

尽管有响应式原型是好的，因为你可以向客户演示站点是如何响应不同视口宽度的，但你必需把控整个陈述过程。在项目早期，应使用你自己的电脑做演示，你肯定不希望在错漏还没有全部解决之前，让客户用他们的手机或设备进行体验。

在项目早期，你可能只是想简单地展示下屏幕截图，这样你就可以专注于网站在主要断点处的显示效果，而不是网站的行为或者交互。

首先从展示小屏幕布局开始，解释这种布局是用户在像手机那样的小屏幕设备以及一些功能有限的老式浏览器和设备上所看到的，然后再转向更大屏幕宽度的布局。

此外，确保早期会议侧重于大问题，而不是小细节。你希望字体大小与布局相适应，而不是围绕客户想要的*确切的*字体大小来构建布局。

让讨论紧扣主题，同时确保你的展示进行地足够慢以便客户能够真正看到所有细节。

记着经常向客户展示设计进度以便不会出现令客户大吃一惊的事。你要避免所谓的“大揭秘”，将设计突然就一下子展现在客户面前。

当你转去展示实时原型，并演示当视口大小发生变化时布局是如何改变的时候，确保客户明白，调整浏览器窗口大小所看到的显示效果与网站在实际设备上观看的效果是有差异的。如果在开会时没有更多的设备可用，你可以用屏幕截图代替。

应明确告知客户设计中的任何事物都是可以改变的，也应该改变，如果感觉它们不是最佳方案的话。在任何时候客户都能随时提供反馈，具有全新眼光的客户往往能发现你的团队从未注意到的错误或问题。

[小贴士]

如果你需要一种向基于 HTML 的原型添加标注的方法，可以试下 Elliance 公司的 Metaframe 工具（https://github.com/elliance/metaframe/），它允许你用一个简单的 CSS 类在页面设计上添加带编号的标注。

7.9 总结

在本章中，我们探讨了在创建网站的视觉设计之前为什么要先从用户研究和内容策略入手的原因。

我们还探讨了设计流程中的各个步骤，先用草图绘制设计理念并以小屏幕优先的视角来进行布局思考。我们谈到了原型，以及响应式原型与传统的线框图的差异。

我们学习了使用样式板呈现网站的视觉观感，以及可以用什么工具来设计响应式网站。

最后，我们讲了如何向客户或同行推销响应式设计，以及如何与客户在响应式项目上合作，以确保客户对他们能得到什么样的网站有一个清晰的认识。

在下一章中，我们将谈谈响应式网站的用户体验，以及移动设备需要你对设计思考方式做出何种改变。

第 8 章

岂止手机

当你创建一个响应式网站时，有两件事需要充分考虑：用户与设备。在本章中，我们将对两者进行全面地讨论。

首先，我们来看看什么是用户体验，它为什么重要。随后，我们将讨论确保网站在尽可能多的设备上有效工作、良好显示的设计策略。

接下去，我们将重温现有的设备类型，以及一些属于重要设计考量的设备特性，比如触摸及屏幕尺寸。我们还将讨论如何确保你的网站对于残障用户也是可访问（无障碍）的。

最后，我们将讨论响应式网站的测试——应该支持哪些设备并用其来测试网站，如何获得这些设备，在没有实际设备时你又可以做怎样的测试。

8.1 用户体验

在制作一个网站时，你的工作并不仅仅是使网站好看 , 而是要使网站能达到预期目标，且运行良好。

电子商务网站可以有非常漂亮的产品页，但如果用户在完成支付过程中遇到困难，并因此去了其他网站，那再怎么漂亮也是没有意义的。

用户体验一直是网页设计的重要组成部分，但很长一段时间以来我们没有为之付出太多的努力，因为所有用户的体验都非常相似，他们都是用的台式机或笔记本电脑，具有类似尺寸的屏幕，使用键盘、鼠标或触控板进行导航。我们的设计也是在同样类型的设备上进行的，所以很容易预测其他用户是如何体验网站的。

但现在一切都不同了。不仅是屏幕尺寸多变，从小不点到巨无霸，应有尽有，而且我们与网站的交互方式也变得多种多样，从触控板到语音控制，各不相同。

我们也已经不再把互联网想成我们体验过的某个事物。过去我们会坐在桌前，打开电脑然后上网。但现在我们总是处在随时联网的状态下，我们可以从口袋中掏出手机随时随地进行各种交互。我们可以把电视或游戏机连上网，甚至在多个设备上同时与浏览器进行交互。

要记住一点，响应式设计并不仅仅是为移动设备进行设计。现在我们倾向于关注移动设备，是因为我们以往都是为非移动设备做设计，自然的，我们需要对移动设备进行更多学习和适应。但仍有很多人是坐在桌旁面对着显示器的，所以你创建的网站不能只仅仅为移动设备进行过优化，而忽略了使用键盘和显示器的人们。

在制作一个响应式网站之前，你需要对浏览它的设备以及用户与这些设备的交互方式进行了解。你的目标是拥有这样一个网站，它能在广泛的现有设备上工作，同时也是未来友好型的，能够兼容于还未被发明出来的设备。

8.1.1　用户至上

哪怕是在我们开始谈论移动设备之前，我们还是需要在用户问题上再多讲一些。毕竟，在你制作一个网站时，用户是方程式中最重要的部分。

HTML5 和 CSS3 支持的新特性加上市场上所有新型移动设备的创新性，很容易使人兴奋得忘乎所以，并花费时间和努力去开发激动人心的网站向世人炫技。

但是请记住，网站的目的并不是要在技术上给人以深刻的印象。学会设计与开发一个网站并不仅仅是编写呈现网站的代码。它是要为使用网站的人们创造一种体验。

差不多所有的网络从业人员都在谈论响应式设计，制作响应式网站和思考响应式设计。我们需要明白，普通的网站用户并不知道什么是响应式设计，也不需要知道。

所有用户想要的是一个能在他们*当时*所使用的设备上良好工作的网站，这个设备不一定与他们在其他时候所用的设备相同。他们并不会去考虑自己在使用什么设备，他们只是想要一个能工作的网站。

用户不会考虑这样一个事实：他们在上班路上使用手机访问网站，在午休时间使用台式电脑，在深夜则边看电视边用平板电脑上网。使用什么设备对他们不重要，他们只要得到一个能工作的网站，而不管在使用什么设备访问它。

因此，当你设计一个响应式网站时，请记住，你的目标不只是创建一个响应式网站，还要创建一个*有效服务于用户的网站*。响应式设计是你将用来创建这样的站点的工具。

用户不会在意网站内部的细节，只是需要一个能给他们想要的内容和功能的网站，能够尽

可能省事地去他们想去的地方。

千万记住，是由用户决定他要使用什么样的设备，而不是由你。你可以决定支持哪些类型的设备，但是如果你的网站不能在用户想要使用的设备上工作，你会失去这些客户。

8.1.2 手机用户的谬见

iPhone 刚一问世，我们也开始设计能在手机上工作的网站，随即产生了这样一个关于“手机用户的谬见”：认为使用手机的人都是“在外面”四处奔走忙个不停，急于去某个地方，急于获得信息。典型的手机用户并不会想要浏览网页，他只是想获得快捷信息，比如一个餐厅的地址或其航班是否准时。

在那一阶段，这一谬见可能是正确的。在智能手机上访问网站的体验很糟。那时还没有响应式网站甚至移动版网站都很少，因此基于台式机屏幕设计的网站在手机上被缩得极小，要想阅读必须不断地放大缩小。因此绝无必要，用户不会用手机访问网站，留着在家中或办公室中用电脑上网显然容易得多。

但那之后网站开始做出改变以适应手机，而且在手机上做事也变得越来越容易。然后平板电脑出现，其算移动设备，但又不全算，因为大多数平板电脑只支持 WiFi，所以你使用它们时并不能真正地在外四处随意走动。

现在人们无时无刻不在使用着手机，不管是在户外还是坐在离笔记本仅几步之遥的家中沙发上，这一切都是因为手机更为易用。

越来越多的人依靠移动设备作为他们的主要或唯一的访问互联网的方式，所以你不能继续假设用户只是想查找餐厅地址或航班时间。人们想要在他们的移动设备上做任何事，就像他们能在桌面电脑上所做的一样，只要你能让他们那么做。

8.1.3 为应用环境设计

虽然我们不应该去假设特定设备的用户只需要某些内容或交互功能，但我们可以确定网站的哪些特定部分在某种设备上用得更多，并去确保这些内容非常容易访问。

例如，我们不能假设移动设备用户只需要某些内容，因为他们不一定是“在移动中”并在找寻“移动型内容”。但不管怎样，他们可能有时是“在移动中”的。并且由于在手机上使用手指来导航网站要比在桌面上用键盘和鼠标导航网站困难得多，我们应该确保移动设备用户能够很容易地获取重要的内容。

图 8-1 所示的响应式网站来自澳大利亚的 Kiwi Bank，在其窄屏设计中页面顶部只包括几个关键的链接及信息。这些链接中最上方的是登录链接，其在宽屏设计中也是突出显示在右上方的。

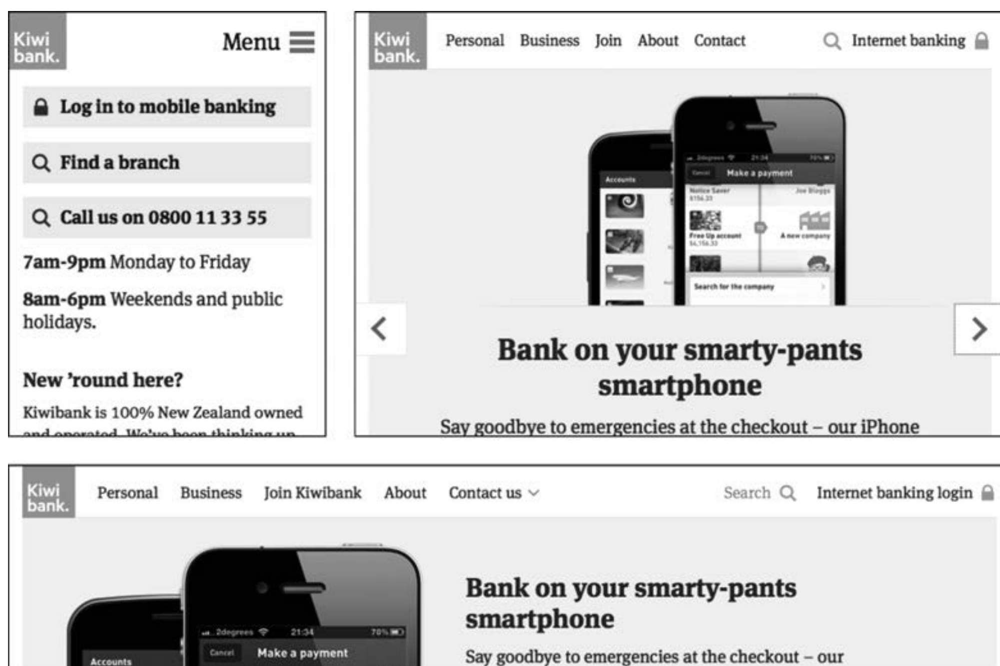


图 8-1：屏幕上最突出的内容可用媒体查询根据视口宽度来改变

紧随其后的是“Find a Branch”（查找分行），银行客服电话，以及银行营业时间。这些内容都是手机用户在忙碌中想要方便并快速获得的。

不过在网站的宽屏版中你看不到“Find a Branch”、电话号码或者营业时间。它们仍是存在的，只是需要点击“Contact Us”（联系我们）才能看到。桌面用户很可能没有那么急切，他不会介意多点几下才能看到这些信息，在有鼠标可用的时候这很容易。

手机用户仍能得到你在宽屏设计中看到的一切，只不过在窄屏设计中需要向下滚动才能看到导航及所有的内容。通常查寻利率的手机用户可能没有查寻分支机构营业时间的手机用户那么急切，因此他对需要向下滚动屏幕才能看到利率信息不太会介意。

8.1.4　只用手机的用户

如果你正在读这本书，那么很可能你有份工作，一天中的大半时间都是在台式机或笔记本电脑前，需要时你就可以上下网。但别忘了不是每个人都在办公室中工作。例如，从事服务工作的人们通常不会把工作时间花在电脑前。他们中的一些人在家里有电脑，另一些则没有。而且有电脑的那些人中也会有一部分更喜欢使用移动设备。

智能手机的崛起意味着越来越多的人通过移动设备上网。按 Pew Internet 的说法，在 2013 年有 57% 的美国人表示他们使用移动设备访问互联网。更出人意料的是这其中有 34% 的

用户表示这是他们的主要上网方式。[1] 这是一个巨大且不断增长的用户群体。

青少年日益成为移动设备的主要用户。虽然他们中的很多人在家中可以使用台式电脑，但他们觉得在手机上有较大的私密空间，手机只属于他们，不像台式电脑那样要与其他家庭成员共享，没有太多的隐私可言。他们也不像许多办公室职员那样整天坐在电脑前。青少年在你的目标受众中可能不太重要，但随着他们长大成人，他们将继续轻松自在地使用手机作为其主要的上网方式。

这意味着什么呢？意味着你的网站必须适用于所有的设备。你不能假设如果某些操作或内容对于小屏幕太“复杂”，用户就会因此切换到大屏幕上去，然后你就可以忽略掉这些问题。但事实上并不是所有的用户都会这样。

并且不要以为只要告诉用户他们需要切换到桌面浏览器就够了。选择什么样的设备不是由你而是由用户决定的。

使任何事物在小屏幕上都能容易使用通常不太现实，但至少不要使之变得不可用。除非，你愿意失去一部分用户或客户。

8.1.5　使用多个设备

在创建能在任意尺寸的屏幕上工作的响应式设计时，需要考虑的一件事是，任何给定的用户可能在不同的时间使用不同的设备访问网站。

虽然你可以用响应式设计改变网站在不同屏幕宽度下的显示布局，但要确保哪怕是在不同尺寸的屏幕上，用户的体验应该是相同的。用户应能在不同的设备上访问网站，而且要让用户确信访问的是同一个网站。不管他们使用的是什么设备，配色方案、图像和字体都应该是相似的。

像导航之类的物件可能根据屏幕尺寸的不同有很大的变化，但可用的选项应该是相同，在导航中应使用相同的信息架构组织它们。

8.2　设备无关的设计

在开始设计响应式网站之前，你应该知道网站会在什么地方被浏览。答案是：任何地方。

现在有数量令人难以置信的设备可用，即使我们开始讨论所有这些设备，我们也不知道接下来又会发明什么新设备。

它们有什么共同之处？

注 1：完整报告请查看 MaeveDuggan and Aaron Smith 发表的“Cell Internet Use 2013”，皮尤互联网研究项目，2013 年 9 月 16 日（http://www.pewinternet.org/2013/09/16/cell-internet-use-2013/）。

它们中的大多数拥有一个屏幕，但并非全都这样。一些失明或视力受损的用户将使用屏幕阅读器上网。具有互联网连接功能的汽车可以为你阅读电子邮件或新闻提要，也许用不了多久就可以在汽车上通过听来上网冲浪了。

人们使用各种各样的输入设备来访问网站。在桌面电脑上，你可能会使用键盘和鼠标。在笔记本电脑上，鼠标被触控板替代。在手机或平板电脑上，你使用触摸屏。在旧手机上，你可能会使用方向按键。

这里的重点是，你不能只为特定类型的设备进行设计，因为有那么多不同类型的设备被创造出来，远远超出了我们对一个标准的移动设备的认知，甚至没有办法预测又会有什么设备将被发明出来，其可能甚至会是我们大多数人无法想象的。

设备无关的设计（device-agnostic design）意味着创建一个不管是在什么类型的设备上都能工作的设计。你不是为移动设备进行设计，也不是为桌面电脑进行设计，你是在为网页进行设计，无论在什么地方在查看它们。

8.3　专注于移动优先

我们在前一章谈到了*移动优先*（mobile first），其理念就是创建一个这样的设计，优先考虑使用移动设备的用户以及这些用户如何与网站进行交互。

当你创建一个设备无关的设计时，这是最好的开始方法，不是因为移动设备上的用户体验比桌面电脑或其他设备的用户体验更重要，而是因为移动设备有更多的限制因素，在屏幕空间有限以及对用户交互方式（触摸）还不太熟悉的情况下更难以创造好的用户体验。

我们已经在第 5 章中谈到了优先为小屏幕设计。但移动优先的理念超出它，它聚焦于用户如何与网站和设备交互。触摸是最大的复杂因素。一个可以很容易使用鼠标导航的布局可能在你用你的大手指在一小块玻璃上戳来戳去时会变得很难用。

在本章后面部分，我们将讨论一些特定的移动设备问题，比如触摸屏和设备性能。而在第 11 章中，我们将谈及性能，它是另一个对手机有普适性的问题，性能的改善将普惠于所有的设备。

8.4　尽你所能

响应式设计不是一个非此即彼的方法。虽然理想情况下你将从零开始为每一个项目创建一个崭新的完全的响应式网站，但在现实中你经常是要使用现有的网站，对其进行修改或重新设计使之成为响应式网站。

如果你正在使用现有网站遗留下来的代码，或者资源有限，不一定能使网站是完全响应式

的。但部分响应式网站也要优于完全不具响应性的网站。

例如，亚马逊网站不是完全响应式的，只有一些元素具有响应性，只出现在更宽的屏幕中（有一个单独的移动版网站用于手机大小的屏幕）。

页面元素有一个固定的宽度，如图 8-2 所示，页面的右边在较窄的浏览器窗口中被切断。

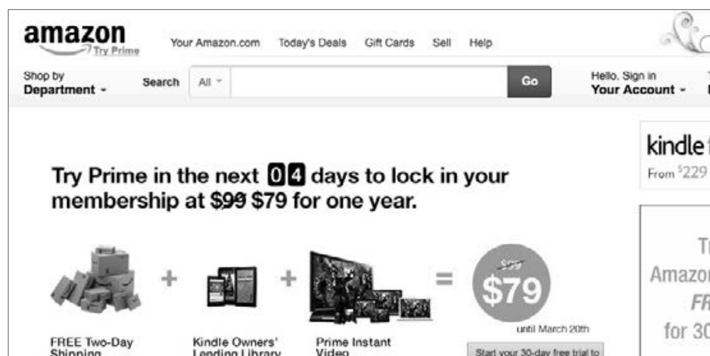


图 8-2：亚马逊网站页面的右边在较窄的浏览器窗口中被切断

但随着视口宽度的增加，如图 8-3 所示，元素之间的空白变大，这样页面能够继续填满整个屏幕的宽度。

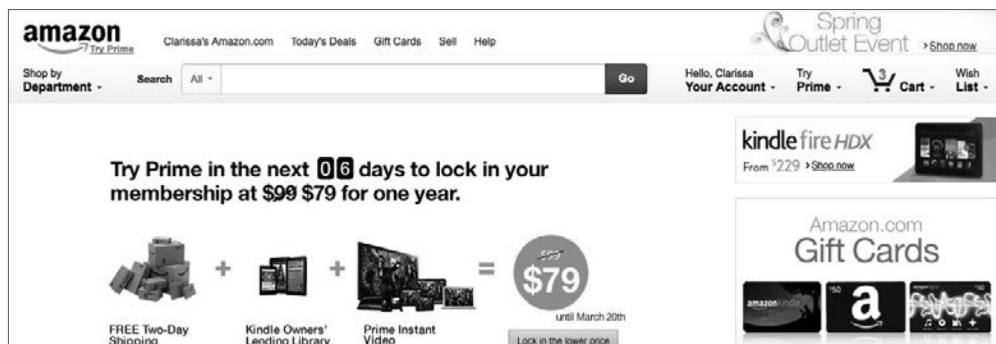


图 8-3：在较宽的屏幕上，元素间的空白变大

在视口宽度足够宽时，因为有足够的空间，子导航菜单“Shop by Department”从下拉链接变成了完全可见的导航栏，如图 8-4 所示。视口变得更宽时，空白进一步扩大以填充页面，如图 8-5 所示。

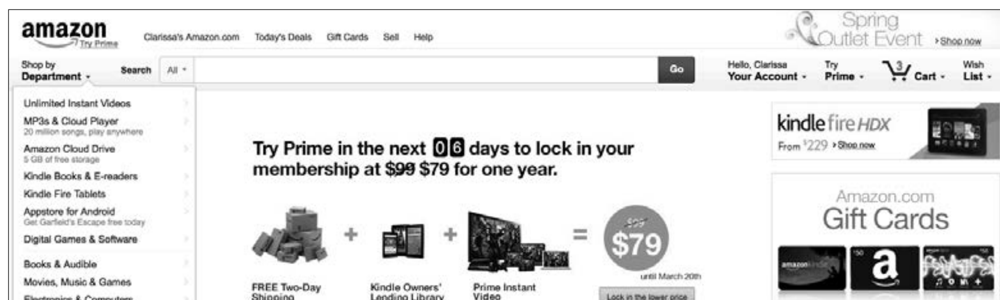


图 8-4：在足够宽的视口中，“Shop by Department” 菜单从下拉链接变成一个完全可见的导航栏

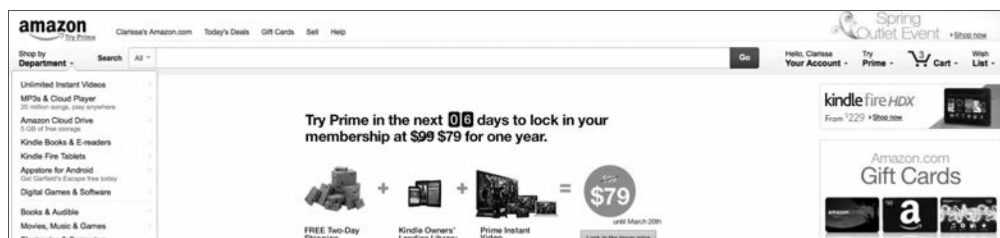


图 8-5：空白进一步扩大以填充页面

在平板电脑上，桌面网站会简单地缩小来准确地适配窗口宽度（如同所有固定宽度网站所默认的那样），如图 8-6 所示，尽管这意味着文本会相当小，在竖排视图中难以阅读。

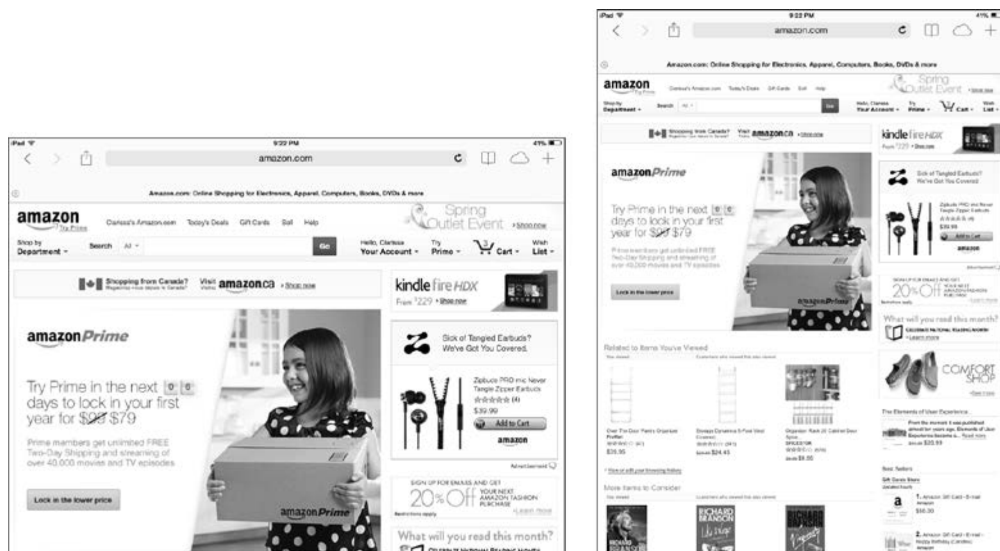


图 8-6：在平板电脑中，桌面版网站缩小以适配视口宽度

这不是一个响应式网站，但它在一定程度上能适应不同的屏幕尺寸。这是具有稍许响应式

性也要好过没有响应性的例子。

8.5 设备类型

有大量不同类型的设备可以上网，从我们熟悉的 iPhone 到所有尺寸的平板电脑、笔记本电脑、台式电脑、游戏机，甚至冰箱。

我们将简略地重温这些设备类型以及它们之间的差异性，然后在本章的剩下部分谈论在设计能于所有类型的设备和所有可能的屏幕尺寸上工作的响应式网站时，需要考虑些什么。

8.5.1 手机

早在 20 世纪 90 年代中期，手机就可用来上网，尽管只能浏览基于文本的网页，这与我们今天所享有的丰富内容相去甚远。直到 2000 年前后，手机上网才变得越来越普遍，虽然还只是个功能有限的版本。

智能手机，比如 iPhone，实际上是一台微型电脑，具有移动操作系统，允许用户安装应用程序。

另一方面，*功能手机*（feature phone）则不太像电脑，功能手机在 iPhone 问世之前被人们大量使用。它们可以上网，但当你查看网站时，网站看上去跟它在显示器屏幕上的样子一点也不像。功能手机比智能手机的功能要少得多，通常也不配备触摸屏，你可能不得不使用如图 8-7 所示的小小按键来在网站中导航。



图 8-7：功能手机（图片来源：Rodrigo Senna，http://www.flickr.com/photos/negativz/38422354/）

但功能手机并没有完全被智能手机所取代。在美国，58% 的成年人拥有智能手机，32% 的成年人拥有功能手机。[2] 在一些国家智能手机拥有更高的渗透率，比如阿联酋和韩国，都在 73% 左右，而另一些国家，比如撒哈拉以南的非洲国家，功能手机的使用率则要高于智能手机。[3]

响应式设计的一部分是确保网站能在任何一个设备上工作，不仅仅是使之适配屏幕尺寸。如我们在第 3 章中谈到的，从一个好的 HTML 架构开始，并以渐进增强的方式为可支持的设备添加 CSS 和 JavaScript，这样可以确保网站在决大多数只具基本功能的手机上也是可用的。

如果网站有很多复杂的交互，响应式设计可能无法使整个网站都能在所有设备上（包括最老的浏览器和设备）良好工作，但尽你所能让其能在尽可能多的设备上工作。

例如，也许你无法使餐厅网站的在线下单功能可以工作于最老的手机，但你应该使包含餐厅地址和电话号码的页面能够在几乎所有的设备上工作，这并不需要费什么劲。

8.5.2　平板电脑

平板电脑近年来已经变得越来越普及。它们存在多年，但直到 2010 年 iPad 的发布才变得流行起来。iPad 是迄今为止最受欢迎的平板电脑，其他品牌比如亚马逊的 Kindle Fire 被远远地抛在了后面。

平板电脑非常类似于现代的智能手机，通常使用相同的操作系统，比如 iOS 和 Android，允许安装应用程序。

许多平板电脑只能通过 WiFi 上网，而不是通过蜂窝移动网络（cellular network），所以在技术上它们不被列为移动设备。但这其实无关紧要。移动设备与非移动设备的分界线已经变得很模糊，坚持把每件设备按移动与非移动来归类已不再有用。这更像是一个模糊的连续统一体，一端是手机（通常在移动），一端是台式电脑（通常不移动）。

电子书阅读器是主要用于阅读下载的电子书籍的设备，但它们中的许多现在都包含有网页浏览器，并更多的是作为平板电脑销售（比如 Kindle Fire）。某些电子书阅读器，比如更基础版的 Kindle，内置的浏览器功能有限，但使用电子墨水屏代替传统的屏幕，所有的内容都以灰度显示。

8.5.3　其他设备

现在很多游戏机内置有浏览器，只要将屏幕连上游戏机就能上网。

注 2：完整报告请查看 Susannah Fox 和 Lee Rainie 发表的“The Web at 25 in the U.S.”，皮尤互联网研究项目，2014 年 2 月 27 日（http://www.pewinternet.org/2014/02/27/the-web-at-25-in-the-u-s/。

注 3：更多信息请查看 Google 的 Our Mobile Planet（http://think.withgoogle.com/mobileplanet/en/）。

有些游戏机只有非常简单的浏览器，这意味着你在屏幕上看到的内容可能并不完全符合设计师的预期。但与功能手机一样，只要网站始于良好的 HTML 结构，然后添加 CSS 和 JavaScript 做渐进增强，它在这些设备上应该也是可用的。

你永远不知道在哪又会冒出一个浏览器。例如，具有 WiFi 功能的冰箱已经存在数年了（有点像冰箱前面嵌入了一个小平板电脑）。目前，它们所搭载的操作系统功能有限，只支持特定的应用程序，但我想不要太久，在厨房中，你会拥有一个完整功能的网页浏览器。

现在还有手表中嵌入的浏览器，汽车中的浏览器，再加上谷歌眼镜，它把浏览器直接放到你眼前。这些浏览器都有非常不同的界面，你无法预见未来几年会出现哪些可能的设备。但幸运的是，这正是响应式设计的用武之地，你的目标是设计一个未来友好型的网站，能够工作在任意尺寸的屏幕和任何设备之上，无关性能。

8.5.4　台式电脑与笔记本电脑

直到几年前，我们还仅限于用台式机或笔记本电脑上网。这可能是我们所有人最熟悉的配置：

- 大屏幕；
- 使用键盘或鼠标、触控板或轨迹球输入；
- 台式机，在桌上使用，通常是坐在它前面，不太能改变位置；
- 笔记本电脑，可在桌上使用，或坐着 / 躺着时放在膝盖上使用；
- 差不多总是使用有线连接或 WiFi；

我们常常认为这些设备拥有高速连接，但这并不一定是真的。许多家庭的网络连接并没有那么快，此外通常在酒店、咖啡馆或者其他地方使用笔记本电脑时 WiFi 也很慢。有些人在农村地区没有宽带可用，连接的是卫星互联网。

8.6　触摸

除了屏幕尺寸，智能手机与传统电脑的另外一个关键区别是输入方式。

台式机和笔记本电脑习惯上使用键盘结合定位设备（比如鼠标或触摸板）作为其输入方式，我们能够很容易地设计出支持这些设备的网站。支持触摸的网站则更难以设计，要确保良好的用户体验必须付出更多的努力。

你可以对网站进行修改使之能更好地工作于触摸屏，而且这些改变并不会降低非触摸屏用户的体验。

但不要认为你只需要考虑手机和平板电脑的触摸问题。现在，许多投入市场的台式电脑也配备了触摸屏显示器（连我妈都在 QVC 有线购物频道上买了一台）。所以最好假定所有尺

寸的屏幕都是可触摸的。

如果你的网站是为触摸屏而设计的，它必然也能在使用鼠标的情况下有效地工作，但反过来情况就不一样了。

> **[小贴士]**
> 设计出良好的触摸用户体验有一定难度。如果你想对此领域进行深入研究，可参阅 Josh Clark 的《触动人心：设计优秀的 iPhone 应用》(http://shop.oreilly.com/product/0636920001133.do)。尽管此书是关于应用程序设计的，但大多数观念与网站设计是共通的。

8.6.1　电容式触摸

自 iPhone 出现以来，智能手机上的主要输入方式一直是*电容式*触摸屏。

电容式触摸屏意味着屏幕是通过测量物体（比如手指尖）触及屏幕所传导的电荷量来工作的。人体导电，但并不是所有的物体都行，这就是用钢笔或带手套的手指触摸智能手机屏幕没有效果的原因。不过，可以通过购买由导电材料制成的特殊手写笔，甚至是指尖部位是由导电纤维织成的手套来解决这个问题。

老式设备有时用的是电阻式触摸屏，其有两个薄膜层，当用户在外表面用指尖或手写笔按压时薄膜层发生接触，电阻屏就能够确定压力的精确位置。这种类型的屏幕有一个缺点是，需要施以一定量的压力，如果触按力量太轻，屏幕通常没有反应。也正是由于需要施加一定压力，电阻屏也更容易损坏。

8.6.2　多点触摸

虽然 iPhone 于 2007 年推出的时候具有触摸界面的手机已存在多年，但 iPhone 是第一部具有*多点触控*界面的智能手机，其允许屏幕可以同时识别多个触点，这使得人们可与屏幕做更加复杂的交互，比如捏合缩放（pinch-to-zoom）动作。

8.6.3　手势

现代智能手机将*手势*作为用户与屏幕内容交互的一种方式，这给予了用户一种“隐藏”的方法来使用界面，从而减少了控件所要占用的宝贵的屏幕空间。

捏合缩放手势是一个最常见的手势（gesture），允许以一种直观的方式来放大或缩小地图之类的事物。如果是在笔记本或台式机上查看相同的地图，你会看到一个具有加、减按钮的滑块占据了一部分的屏幕。

有些手势，比如放大和缩小网页，是内置于浏览器或操作系统中的。其他的手势，比如在

幻灯片中向两侧滑动屏幕以切换图像的手势，则必须通过添加 JavaScript 才能实现。

虽然你可以跳过这一步，要求用户通过轻触或点击箭头来在幻灯片中切换，但加上手势支持可以使网站更容易在触摸屏上使用。为支持手势的设备添加额外的手势，能增强用户体验，这是响应式网页设计的一个关键部分。

有数个 JavaScript 插件可用来添加滑屏行为，比如 TouchSwipe（http://labs.rampinteractive.co.uk/touchSwipe/demos/）或 Swipe（https://github.com/thebird/Swipe）。

8.6.4 JavaScript事件

触摸事件（touch event）是 JavaScript 中的一个用语，用来描述用户与屏幕做出的交互，是简单的触按还是多个指头移动。有三个基本的触摸事件：`touchstart`、`touchmove` 和 `touchend`，可用于定义几乎所有的触摸交互行为。

此外，触屏设备能够识别为鼠标设计的 JavaScript 事件，比如 `click`（点击）事件。

1. 悬停

在触摸屏上处理 `hover`（悬停）事件有点棘手，很显然，你不可能只是把手指悬空停在元素之上。

悬停在网站上多用于通过 CSS 改变元素的可见性（隐藏或显示），比如当你的鼠标光标悬停在菜单栏的一个链接上时显现一个下拉菜单。

大多数触摸屏通过将元素的行为改变成双触击自动适应此事件。这意味着第一次在元素上触击视作一个悬停动作，并改变元素的可见性；而第二次触击则选中链接。不过，这是种笨拙的实现方式，应该避免使用 `hover`。

当然，你可以使用媒体查询使得较小的屏幕拥有不同类型的导航（见第 10 章），但现在很多台式电脑也配备有触摸屏，所以你需要假设任何尺寸的屏幕都可能是触摸屏。

2. 触摸延迟

你可能已经注意到，同样是触摸，移动应用程序的响应速度有时似乎要比在手机中浏览网站的反应速度快得多。导致此结果的原因实际在于浏览器内置的 JavaScript 引擎在触摸设备上特殊的工作方式。

手机浏览器中的 JavaScript 引擎会尽力复制鼠标事件，将 `onClick`（鼠标点击）事件融合到 `touch`（触摸）事件中。因此如果浏览器监听一个元素上的 `onClick` 事件，它知道在触摸屏上一次触击即一次点击，但又有点不同，浏览器会等待并查看你是否会再次触击，从而形成两次触击（双击，将放大网页）。也就是说，在你第一次触击后，浏览器会等待 300 ms（0.3 秒），在确保不是双击后，才形成鼠标单击（click）事件。

三分之一秒看似不多，但对于用户，有或没有 300 ms 的延迟，差别还是很明显的，因为这意味着他们在触击目标时事情似乎并没有“马上”发生。

还好，至少有一个浏览器厂商意识到这个问题并开始修复它。新的 beta 版的 Chrome for Android 浏览器移除了 300 毫秒的延迟，但只对视口（`viewport`）元素标记的 `content` 属性设置为 `width=device-width`（`<meta name="viewport" content="width=device-width">`）的网站，即响应式网站有效。当然用户也无法再通过双击来放大网页，但缩放在响应式网站中并不是那么必需，况且用户仍可以捏合手势来进行缩放。

目前还不清楚其他浏览器厂商是否会在这方面跟进，不过在此期间可以用 JavaScript 插件来移除网站上的 300 毫秒延迟。查看 FT Labs 的 FastClick（https://github.com/ftlabs/fastclick）或 Ben Howdle 的 Touche.js（http://benhowdle.im/touche/）。

8.6.5 触摸目标大小

对于使用触摸屏访问网站的用户，最大的一个可用性问题是触摸目标的大小（即，链接或其他可通过轻触屏幕来选取的元素的大小）。

当使用鼠标点击链接时这不是那么重要，因为用户（对于大多数用户来说）能很容易精准地移动光标至正确的位置并点击链接或其他元素。但对于触摸屏幕的用户，要知道手指比光标大得多，且如果屏幕上触摸目标确实很小的话，很难触及正确的位置。

如果一个目标独占一块区域，问题不大，但如果目标是非常紧密地挨在一起，一不小心很容易误触其他目标，把你带到你不想访问的页面，或执行一个你不想执行的动作。

你会发现在使用触摸屏时，用户实际上倾向于使用他们的手指肚（指纹所在处），而不是手指尖（手指最前端）。因人而异，这实际上可以是相当宽的区域。

如果用户试图触及一个非常小的目标，有时可能用指尖而不是指肚会更准确些，但这会降低用户的操作速度并增加操作难度，这不会是你想要的结果。

根据麻省理工学院触摸实验室（MIT Touch Lab）的一项研究，食指的平均宽度是 1.6~2cm（0.6~0.8 英寸），转换成像素大概是 45~57 像素。幸运的是，智能手机非常的聪明，当它收到触摸事件时，即使你的手指在屏幕上触摸目标时可能有些许误差，它通常可以判断出你意图触及的目标。但如果目标太小，成功的机率将会下降。

手机厂商实际上会有相关的设计指南，并在其中给出一个元素应该多大的建议，从而使用户可以在触摸屏上很容易地选择触摸目标。苹果的 iPhone 人机界面指南建议的触摸目标的大小至少是 44 像素 × 44 像素。微软 Windows Phone 的建议是 34 像素宽，最低不少于 26 像素。诺基亚建议触摸目标应该至少是 28 像素 × 28 像素。

这么多不同的建议，用哪个好呢？如果触摸目标是一个紧挨着一个，我建议在可能的情况下应使它们至少是 44 像素 × 44 像素。

当你在各种移动设备上测试你的网站时，务必检查链接在触摸屏上是不是容易选择。记住，别人可能不如你的手指纤细灵活，某些人的手指会更粗大些，年长用户可能更难以触及屏幕上精确的点。

加大触摸目标

你可以使用 CSS 来增加触摸目标的大小。方法之一是确保链接周围的空白是由*内边距*而不是*外边距*产生的。你需要回顾下在第 3 章中学过的盒模型，其解释了内边距与外边距之间的区别。

考虑一下图 8-8 中的这个链接列表。链接挨得很近，取决于文字的大小，可能难以用手指正确触碰。我用线勾勒出了每个链接，这样你可以看到哪一部分是可点击 / 触碰区域。

- Lorem ipsum dolor sit amet.
- Lorem ipsum dolor sit amet, consectetur adipiscing elit.
- Lorem ipsum.

图 8-8：没有应用样式的列表中的链接挨得非常紧密

我们对此可做哪些改进呢？一开始，你可以用 `display: block` 把链接变成块级元素，这将使 `<a>` 元素框（盒子）延伸至整个包含元素的宽度，如图 8-9 所示，而不是停止在最后一个字符的末端：

```
ul a { display: block; }
```

- Lorem ipsum dolor sit amet.
- Lorem ipsum dolor sit amet, consectetur adipiscing elit.
- Lorem ipsum.

图 8-9：把链接由行内元素变成块级元素，使得触摸目标框扩展到整个包含元素的宽度

要加大可触区域，你也可以加大每个 `<a>` 元素的内边距（不是外边距），你可以任意选择在每个方向如何加大触摸目标的大小，如图 8-10 所示：

```
ul a { display: block; padding: 3px 5px; }
```

- Lorem ipsum dolor sit amet.
- Lorem ipsum dolor sit amet, consectetur adipiscing elit.
- Lorem ipsum.

图 8-10：添加内边距可在任一或所有方向上加大触摸目标的大小

不要对 <li> 元素应用内边距样式，否则将留出空档，如此造成的链接间的空档是不可触及的，如图 8-11 所示：

```
li { padding: 3px 5px; }
```

- Lorem ipsum dolor sit amet.
- Lorem ipsum dolor sit amet, consectetur adipiscing elit.
- Lorem ipsum.

图 8-11：如果你在列表项上使用内边距，链接之间留出了的空档，并使触摸目标变得更小

你应该充分利用所有可用的空间，不需要留出空档来分隔链接。鼠标用户会直接点击链接，因此不用担心会产生混淆。

对于文本块（比如段落）中的链接，则有点麻烦。通过确保各行有足够的高度，可以使链接相距得更远一点（我们将在第 9 章看看如何用 CSS 的 `line-height` 属性来做到这一点）。尽量不要使相邻的单词是不同的链接。那样不仅使它们更难点击，，而且在触摸屏上用户可能无法分辨存在有多个链接，因为他们不能够像使用鼠标那样把鼠标光标悬停在链接上来查看链接的目标指向。

另外，请记住在最小的屏幕中这么做可能会相当的棘手，因为屏幕太小没有多少可用的空间。如果你想使触摸目标在小屏幕上更小一些，可以使用媒体查询来改变不同屏幕尺寸中触摸目标的大小。在前面的例子中，内边距分别是 3 像素和 5 像素，这并不适合于小屏幕，你可以使用媒体查询为窄屏设置稍微小点的内边距，而对于平板尺寸或更宽的屏幕则设置更大些的内边距。

8.6.6　导航位置

虽然在网站和桌面应用程序上导航自然而然地演变成出现在屏幕的顶部，通常与首页按钮或搜索框等关键元素一道放在顶部的角上，但这种布局实际上仅在使用鼠标时是最优的。

对于触摸屏，我们需要进行全新的考虑。

其中的区别于在智能手机上使用的应用程序，其导航通常出现在屏幕的底部，使得你能够更方便地用拇指触击这些按钮，如果你是单手握持手机。

不仅仅是手机，对于几乎所有的触摸屏，离手最近的屏幕部分最容易触摸到的，即使在桌面触摸屏显示器上，用户触碰靠近屏幕底部的元素也是最容易和最轻松的。

把控件放置到屏幕的底部并不仅仅是使他们更容易触及，还有一个额外的效果：你的手指（或手臂，对于桌面触屏显示器来说）不会挡住屏幕的其他部位，因此你仍能看到屏幕上的内容。

如何设计导航的位置是一个难题，所以我们在本书中着墨不多。在第 10 章中，我们会看到一个网站导航在屏幕的底部的例子。尽管这种对标准用户界面的调整并不常用，但随着 Web 的使用越来越远离传统的键盘和鼠标，这种做法一定会变得愈加普遍。

如果你想了解更多关于为触摸屏优化导航的理念，查看 Josh Clark 的博客文章“New Rule: Every Desktop Design Has To Go Finger-Friendly”（http://globalmoxie.com/blog/desktop-touch-design.shtml） 或 Luke Wroblewski 的“Responsive Navigation: Optimizing for Touch Across Devices”（http://www.lukew.com/ff/entry.asp?1649）。

8.7 屏幕尺寸

在做响应式设计时，一个主要的考虑是确保网站在任意尺寸的屏幕上都能被浏览。

过去，你可以按屏幕尺寸给设备分类，并设计了几组尺寸：手机、平板电脑、笔记本电脑、宽屏幕。但是现在，设备不再能够这样轻松地分类了，因为它们几乎具有任何能够想象的宽度。

智能手机最小的屏幕尺寸大概在 3 英寸左右（对角线），有许多是在 4~5 英寸之间。目前，最大尺寸的手机是三星 Galaxy Mega，具有 6.3 英寸的屏幕。当你拿着它举到耳边时，它看上去真是硕大无比，不过没关系，很多人用手机主要是为了上网，而不是为了与人交谈（不管怎样，你总是可以使用耳机来打电话）。

iPad mini，平板电脑中的小个子，屏幕尺寸 7.9 英寸，并不比最大的电话大多少。许多平板电脑在 7~10 英寸范围内，iPad 是 9.7 英寸。

触屏超级本的屏幕尺寸从 11.6~13.3 英寸不等，触屏桌面显示器则大很多，我曾看到一台 55 英寸大的触屏桌面显示器在网上销售，不过我相信超过 5000 美元的标价不会有多少人能承受得起。

所以，当你创建一个响应式网站时，需要创建一个设计能适用于所有可能的屏幕尺寸。几年前，为此创建四组设备尺寸的布局就能够工作得相当好了，因为几乎每一部手机都是 iPhone，几乎每一台平板电脑都是 iPad。

但是现在，各种尺寸的屏幕层出不穷，只关注小范围的屏幕尺寸意味着忽略了大量用户。

旋转

一个常见于移动应用程序（移动版网站略少见）的问题是它们不能旋转。

网站应该能旋转以便不管是在水平（横排）还是垂直方向（竖排）都是可用的。诚然，只要按照一个方向进行设计确实不错，但是对于侧滑全键盘手机却有个问题，如图 8-12 所示。



图 8-12：对于侧滑全键盘手机，网站必须要能在水平方向上使用

在侧滑全键盘手机上，只能以水平方向查看应用程序和网站（除非你不需要使用键盘）。

对于用户来说，当他们以自己喜欢的方向握持手机访问网站时，却被告之网站不支持是令人沮丧的，如图 8-13。再一次重申，如何选择及握持设备由用户做主，而你需要做的是使网站做出响应以适应用户。

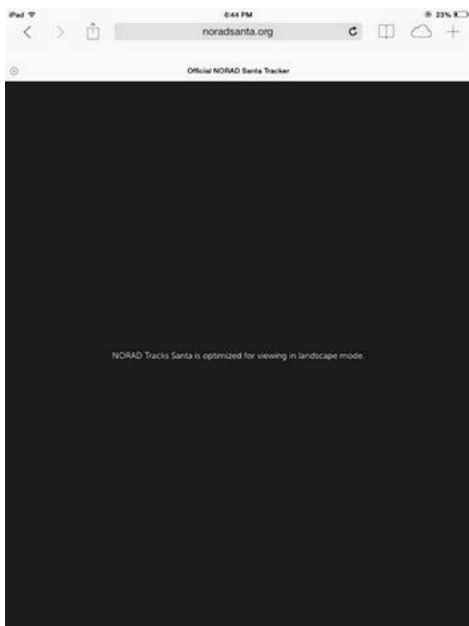


图 8-13：某些网站需要以特定的方向才能查看

8.8　可访问性（通用化设计）

通用化设计（universal design）的概念是资源可以被尽可能广的用户群体所使用，而不仅仅是对普通用户可用。

这其中包括可访问性（accessibility）的理念，其主要含义是确保网站对身体残疾和有认知障碍的用户也是可用的。此外你还要确保网站对老年用户，母语与网站所使用的语言不同的用户以及那些使用过时的或者非常规的设备访问网站的用户都是容易使用的。

不过通用化这个词暗指所有人，事实上如果你使网站对于残障人士更具可访问性，通常你也就使网站对所有人更具可访问性。

纵观本书我们已经解决了可访问性，它在整个网页设计的过程中应明确成为一个考虑因素。但现在我们将讨论你需要关注的各种类型的网页访问方式，以及一些值得思考的关键点。

[小贴士]

可访问性涵盖了很多内容，但由于本书篇幅有限，就不在此展开。想要更多地了解可访问性，可查看 Shawn Lawton Henry 所写的 *Just Ask: Integrating Accessibility Throughout Design*（http://www.uiaccess.com/JustAsk/），其可免费在线阅读；或 Katie Cunningham 的 *Accessibility Handbook* 一书（http://shop.oreilly.com/product/0636920024514.do）。

8.8.1 视觉

因为我们大多数人都认为使用电脑或移动设备主要是一种视觉体验，由此显而易见，对于使用电脑或移动设备一个主要的障碍是看不见屏幕，或者是看不清楚。

1. 屏幕阅读器

举个例子，盲人用户使用称为屏幕阅读器的软件来上网，这种软件向用户大声阅读电脑屏幕上的文字。

使用正确、语义化 HTML（无论如何你都应该这么做）是使网站可访问的第一步。如果使用语义化 HTML 编码网站，正确标记标题、导航和表单等元素，这将使用户能够更容易地导航网站，即使是看不见网站的用户。

在本书中我们已经几次提及屏幕阅读器，并在其他章中讨论了要确保网站对使用屏幕阅读器的用户是可访问的所要做的几件事。

为了更好地了解盲人是如何通过听来访问网页的，可下载一个屏幕阅读器试用下。比如 Fangs Screen Reader Emulator（https://addons.mozilla.org/en-us/firefox/addon/fangs-screen-reader-emulator/），一个由 Peter Krantz 开发的 Firefox 浏览器附加组件。

> [小贴士]
>
> *想了解使用屏幕阅读器的真实体验？你可以看看 David Ball 的博客文章“Things I learned by pretending to be blind for a week”（http://blog.silktide.com/2013/01/things-learned-pretending-to-be-blind-for-a-week/），文中讲述了他作为正常人戴着眼罩使用屏幕阅读器上网一周的体验。*

2. 文本大小

对大多数网站来说，虽然盲人用户只占到受众的很小一部分，但低视力用户则是一个更大数量的用户群体。医学上对低视力的定义是指即使带矫正镜片也不能在正常的视距上阅读报纸的人。为了本书的需要，我将其范围扩大了一点，把它定义为上网时看不清电脑屏幕的人。

这所包含的人数将大大超过你的想象。

当我们变老时视力会自然退化。尤其是，在 40 岁左右时我们的视觉敏锐度开始下降（因此很多年长者戴上了老花镜）。

这意味着许多用户，特别是，但又不仅仅是，老年用户很难看清电脑屏幕上的内容。走过任何一个有年长员工的办公室，你会看到这之中的有些人是眯着眼睛看屏幕，身体前倾试图看清过小的字符。

所以在创建网站时要考虑到此种情况：确保所有文字都是易于阅读的大小，不只是为了你，而是为了大多数用户。我们对此将在第 9 章中做进一步讨论。

还请确保你的代码不会做任何阻止用户调整页面上文字大小的事情，例如不在元视口标记上使用 `user-scalable=no` 属性，我们在第 3 章已经讲述过这个例子。

3. 颜色对比

网站中文字的颜色对其易读性有显著的影响。

白底黑字是简单直接且经典的对比色，但是你可能想使文字颜色的对比变得更有趣一点。

不过，需要记住，如果文字的背景色和前景色没有足够的对比度，访问你站点的用户可能在阅读文字时会有困难。这将包括那些视力不好的用户，比如许多的年长者以及那些色盲患者。即使对于视力良好的人，当试图在刺眼的光线下阅读小小的手机屏幕时，低对比度的文字可能也是一个问题。

记住网站的视觉设计是建立在你的内容基础之上的，而不是要压制内容。

网页内容可访问性指南（Web Content Accessibility Guidelines，WCAG）建议文本颜色与背景色的对比度至少是 4.5∶1。对于大号文本，比如标题，对比度则只需为 3∶1，因为即使是低对比度，大号文字也更容易阅读。有关于此的更多信息，参见 W3C 网站上的“Understanding WCAG 2.0 - Contrast (Minimum)”（http://www.w3.org/TR/UNDERSTAANDING-WCAG20/visual-audio-contrast-contrast.html）。

只通过眼睛来看，通常分辨不出颜色对比是否是可接受的，但是你可以使用在线工具，比如 WebAIM 的 Color Contrast Checker（http://webaim.org/resources/contrastchecker/）来检查两种颜色之间是否有足够的对比度。只需输入两种颜色的十六进制编码，它就会告诉你两种颜色的对比度以及在标准文本和大文本中其对比度是否能通过标准。

另一个工具是 Jonathan Snook 的 Color Contrast Check（http://snook.ca/technical/colour_contrast/colour.html）。输入颜色编码后，它可以让你通过用滑块调整 RGB 设置——色调、饱和度和颜色值，更直观地获得一组合格的颜色。

如果要一次同时检测网站上的的所有内容，可以使用 Giovanni Scala 的 CheckMyColours（http://www.checkmycolours.com）。它会测试网页上的每个元素，并给出一个检测结果表，列出通过检测和未通过检测的颜色。

即使灰色也可能成为一个问题。现在流行用灰色代替黑色来显示文本，它看起来更优雅，但通常难以阅读。

图 8-14 中的文字分别是纯黑色（#000000）和两种不同深浅的灰色（#777777 和 #AAAAAA）。两种灰色的对比度测试都是不合格的。

Lorem ipsum dolor sit amet, consectetur adipiscing elit

Lorem ipsum dolor sit amet, consectetur adipiscing elit

Lorem ipsum dolor sit amet, consectetur adipiscing elit

图 8-14：两种不同深浅的灰色文本未能通过对比度测试

4. 色盲

在网站上运用颜色时另一件要考虑的事情是确保不会对色盲用户造成障碍。

色盲患者并非看什么都是不同程度的灰色，这种类型的色盲是非常罕见的。相反，他们通常是分辨不出某些颜色。例如，最常见的是红绿色盲，对他们来说深浅不同的红色和绿色看起来十分接近。

这意味着对他们来说红绿灯的红灯和绿灯都是相似的颜色。但色盲的人仍可以开车，因为他们知道最上面的总是红灯，最下面的总是绿灯，即使颜色看上去是一样的。在这种情况下，位置被用来传达含义，作为颜色的一种补充。

你需要为网站做同样的事情。将颜色用于视觉装饰完全可按照你自己想法，但如果是使用颜色来传达意义，就需要包含一个补充方法。

例如，你可能会有一个产品特性对比图 / 表，其中使用了红色和绿色去标注是否具备某一功能。与此同时，你还可以 × 和✓来传达同样的信息。颜色看起来引人注目——也许一眼就能看出你的产品绿标远远超过了竞争对手的产品，但无法区分颜色的人仍能够用 × 和✓来理解这个对比图。

当颜色是网站内容的一个关键部分时，你需要提供足够的信息以便用户可以做出准确的选择。

例如，一个销售衬衫的在线购物网站应确保标注出颜色名称，而不是仅仅提供一个颜色方块来演示颜色。此外，还要使用准确的颜色名。假如一个流行的服装网站使用“sugar coral”（糖珊瑚—珊瑚红）和“cloudburst”（暴雨云—蓝铅灰）之类的颜色名。虽然它们让服装听上去很迷人，但那些无意义的名字肯定不会对无法区分颜色的客户有所帮助。

欲了解更多有关色盲对网站用户影响的信息，可以查看 Geri Coady 发表在 24 Ways 上的博客文章“Colour Accessibility”（http://www.24ways.org/2012/colour-accessibility/），文中有很多极好的例子说明了如何确保你的网站符合可访问性标准。

8.8.2 音频

大多数网络内容是视觉内容，还有一些则是听觉内容，比如视频和音频剪辑。对于聋人或

听力不好的用户，这些信息需要以不同的形式出现才能使其具有可访问性。

视频应尽可能配上字幕。遗憾的是，虽然视频创作并不需要多少专业技能（几乎任何人都可以做），但添加字幕就困难了些。如果不能给视频加上字幕，可以选择提供一个文字副本。但还是比不上字幕，因为用户可能无法将视觉图像与文字同步对应上，但这总好过没有。

当然，音频剪辑也应该有一个文字副本。

这不仅能帮助听不见内容的用户。其他用户也可能喜欢阅读文字（这样更快）而不是听。或者他们所在的地方非常嘈杂，难以听清内容，或者是在图书馆那样安静的地方，他们不能大声地播放音频，也没有耳机。

8.8.3　输入方法

尽管有视觉障碍的用户是最明显的需要使用辅助设备上网的一类人，但也有人没有视力问题，但却可能因别的身体原因，不能像我们大多数人那样使用键盘和 / 或鼠标。

1. 只用键盘

一些用户能够使用键盘，但无法使用像鼠标那样的定位设备。这可能是由于患上了诸如腕管综合症这类常见的病。

盲人也属于这一类用户，他们可以凭触觉使用键盘，但无法使用定位设备，因为他们看不到屏幕上指针的位置。

还有些高级用户喜欢用键盘代替鼠标，只是因为他们觉得敲键的速度要快过来回移动鼠标的速度。

默认情况下，HTML 就能使网页对于只用键盘的用户具有相当的可访问性。但你需要确保不会做错事情而破坏这种可访问性。

只用键盘的用户使用 Tab 键（制表键）在屏幕上的元素中导航。从一个可被选择的元素跳到另一个可被选择的元素（表单域、链接等），所选中的元素是突出显示的。试试去一个网站，按几下 Tab 键，你会看到，页面上的链接在选中时是被一个虚线框围住的。如果你按下回车键，产生的效果与点击选中的链接是一样的。

制表键跳转的顺序与 HTML 定义元素的顺序相同，这是确保 HTML 按逻辑顺序编写的另一个好理由。

对于使用键盘导航且看不见屏幕的用户，元素需要标注清楚。比如，对于链接应该描述点击它之后会发生什么，而不是使用千篇一律的“点击这里”。

2. 语音识别软件

完全无法使用键盘的用户可以使用语音识别软件，它使用户能通过语音来指挥电脑。

一般来说，如果网站对于只用键盘的用户是可访问（无障碍）的，那它对于使用语音识别软件的用户也是可访问（无障碍）的。但一个额外的考虑因素是要确保表单域和链接有独一无二的标签，使得语音用户能够轻松地选择他们。

8.8.4 认知障碍

有认知障碍的人是网站用户基数中一个常被忽视的部分。

如我们在第 2 章中讲过的，采用简单明了的语言是使网站对这些用户更具可访问性的一个很好的开始。这也能帮助那些母语与网站所使用语言不同的用户。

你还可以通过在整个网站中采用连续一致的设计模式来帮助用户访问，比如在每个页面上使用相同的导航选项。如果你重新设计一个网站，某些用户将不得不“重新学习”如何使用网站，对他们来说，这可不是简单的找找看的事。

另一个问题是动画图形，其会分散有阅读障碍或多动症（ADHD）用户的注意力。在默认情况下最好不要在网页上有动画图形，并确保不使用动态 GIF 来传达信息，因为有些用户会使用浏览器设置完全关闭掉动画选项。

8.9 决定支持哪些设备

响应式设计的棘手之处在于不只是为不同屏幕尺寸进行设计。毕竟，你可以通过改变浏览器窗口的大小来轻松地设计和测试媒体查询。困难的部分在于如何确保网站能够在尽可能广的各色设备上工作。仅仅是在调整浏览器窗口大小时看到网站显示得还不错并不意味着问题不会突然出现在某些设备上。事实上各浏览器所支持的功能并不完全相同，每个设备和浏览器都可能有小怪癖。

当我们开始一个项目时，我们需要决定站点设计将支持哪些设备，这并不是说网站只有在这些设备上才能工作，只是意味着你仅允诺在这些设备上进行测试来确保网站正常工作。

而响应式网站是被设计成可在任何尺寸或类型的设备上工作的，在这些设备上偶尔发生的怪异情况可能会导致某种错误的发生，而你又不可能去实地测试每一个设备。所以，你需要选择一系列的设备和浏览器（通常能够代表常见设备、操作系统、浏览器、视口大小）在它们上面进行测试。

下面是你可以包含在网站项目计划中的设备及浏览器清单列表。

- 运行 Microsoft Windows 7 及其更新版本的电脑

- Internet Explorer 9 及其更新版本
    - 最新版本的 Firefox、Safari 和 Chrome

- 运行 Mac OS X 10.7 Lion 及其更新版本的电脑
    - 最新版的 Firefox、Safari 和 Chrome

- 运行 Android 4.0（冰淇淋三明治）及其更新版本的平板和智能手机
    - Nexus 7 平板上的 Chrome
    - LG 和 Motorola 智能手机上的 Chrome

- 运行 iOS 6.0 系统的 iPhone 3GS
    - Mobile Safari

- 运行 iOS 7.0 系统的 iPhone 4s 和 5、iPad 2 和 iPad mini
    - Mobile Safari
    - Chrome for iOS

列表中的具体设备将取决于项目的需要。例如，上述列表不包括黑莓或 Windows Phone 设备。现有网站的统计数据可能会告诉你，只有极少的访客使用黑莓手机，所以你可能觉得不值得花时间提供对这些设备的支持。但请记住，如果现有网站不能在某些移动设备上很好地工作，统计数据可能无法表明有多少拥有那些设备的用户因无法使用而离开了你的网站，如果网站能在他们的设备上正确工作，他们是有可能变成正式用户的。

你还应该找出所有利益相关人所使用的设备，并确保在这些设备上进行了测试，即使它们不太常见。不管清单上列出了什么设备，你的利益相关人都会希望网站能在他们自己的手机上正常工作。

8.10　为何要用真实设备进行测试

无需因为你正计划要支持所列清单中的特定设备，就必须实际拥有这些设备。你可以使用在线仿真器和模拟器来查看网站在各种设备上是个什么样子（更多细节在下一节展开）。

不过，尽管在桌面浏览器中使用仿真器能够让你查看网站在各种设备上的显示效果，但它却无法让你去感受网站的用户体验。比如，触摸目标足够大吗？它在各内容部分之间移动容易吗？在慢速的移动网络连接上网站的加载速度足够快吗？

如果你不能获得所有想要测试的真实设备，最起码你应该在一部 iPhone、一部 Android 手机、一台 iPad 和一台 7 英寸平板电脑（iPad mini 或 Android 平板）上进行测试。这是除了在 Mac OS 和 Microsoft Windows 这两个操作系统上的所有主流浏览器之外要进行的测试。

8.10.1 设备实验室

在理想状态下，每个制作网站的公司都将有一个*设备实验室*（device lab），里面有各种各样的移动设备，设计人员和开发人员可以用它们来测试网站。但事实却是，除非你在一个大公司里工作，具有很多的设备预算，否则你不可能那么幸运地拥有这么多测试设备。

你或许可以问问周围的人，至少能找到几个会借给你手机的同事，但很可能他们拥有的设备相似，所以你无法找到所有你想要测试的设备类型。

在过去几年中新出现的*开放式设备实验室*也是一个解决方案。你可以在开放式设备实验室中找到许多开放给社区使用的移动设备。一些设备实验室属于网络机构或其他慷慨的向公众开放其实验室的公司，另一些则是由社区组织或者非营利组织依靠捐赠的资源建立起来的。

世界各地都有开放式设备实验室，虽然大多数都位于欧洲和美国的主要大城市。你可以在 OpenDeviceLab.com 上找到一个设备实验室列表（http://opendevicelab.com）。

你会发现不同设备实验室里的设备各有不同。一些实验室可能只有少量设备，而有些实验室则可能会有大量的不同操作系统和型号规格的设备。一些设备实验室收费少量的使用费用，但大多数都是免费的。

8.10.2 购买设备

虽然购买设备用于测试听起来花费不菲，但还是有办法来便宜地获得其中的一些设备。

必须为每部手机支付合约计划贵得让人望而却步，但是许多手机可以在 Wi-Fi 上使用而无须为合约计划掏钱。这对于测试来说通常已经足够了，尽管它不能让你在移动网络连接上的测试性能。

你可以从 eBay、Craigslist 或其他销售网站获得许多二手旧手机，甚至向朋友和同事询问。并非所有的移动运营商都提供有以旧换新的“抵价”服务，所以人们往往只是把他们的旧手机收入橱柜的盒子中，因为将原本昂贵的设备扔掉似乎让人有些不舍得。

你可能已经拥有或者可以轻易借到一部 iPhone 和 Android 手机，你也值得拥有一台 7 英寸和一台 10 英寸的平板电脑在手边以备测试，这样你可以真实地体会到在这些设备上浏览网站与只是在仿真器中浏览它的感觉是截然不同的。

8.11 测试

正如我们在书中提到的，成功的响应式设计的一个关键是在各种设备上测试你的网站。除了使用实际的设备之外，还有其他的方法来测试你的网站，从而确保它能在最广泛的设备上正确地工作。

8.11.1 验证器

测试网站首先要做的事情之一是验证代码。这能从源代码中迅速捕获代码错误，并能在排除内容显示不正确的故障中节省大量的时间。

验证工具通常内建于主流浏览器的 web 开发人员附件中，或者你可以使用线上服务来验证你的代码，比如可分别验证 HTML 和 CSS 的 W3C Markup Validation Service（http://validator.w3.org）和 W3C CSS Validation Service（http://jigsaw.w3.org/css-validator）。

8.11.2 在浏览器窗口调整

当你设计或编码响应式网站时，你肯定会调整浏览器窗口的大小来查看设计在不同的视口宽度下的显示效果。虽然这是有用的，而且你也应该从那开始，但它不一定能准确反应出网站在实际设备上的真实显示效果。

不过，调整浏览器窗口大小来查看设计这种方法能够给予你一个在移动设备上无法获得的优势：你可以在每个可能的视口宽度下查看设计的显示效果，而不仅仅是你要测试的那些设备的宽度。不管你有多少设备可供测试，你都不可能查看每一种可能的设备宽度，所以浏览器窗口也许是唯一能够看到只出现在 643 和 647 像素之间的某些奇怪的布局的地方。

8.11.3 浏览器工具

当你开始在浏览器中进行测试时，有几个工具可以使测试工作更容易些。

改变窗口大小不但能让你了解设计在所有视口宽度下的显示效果，你还会发现它可用来查看网站在常见设备的精确尺寸下的显示效果。

这在 Firefox 中很容易做到，它有一个内置的工具，可通过“工具”→“Web 开发者”→“响应式设计视图”菜单打开它，如图 8-15 所示。在载入一个页面之后，你可以用下拉菜单在几个不同的预置屏幕尺寸间切换，甚至还提供有按钮允许你对网页进行截图。

在 Safari 中，有一个叫 Resize 的扩展（http://resizesafari.com/）能够让你将浏览器窗口大小调整为可配置的预置大小。

对于 Chrome，可以尝试 Windows Resizer （https://chrome.google.com/webstore/detail/window-resizer/kkelicaakdanhinjdeammmilcgefonfh）。

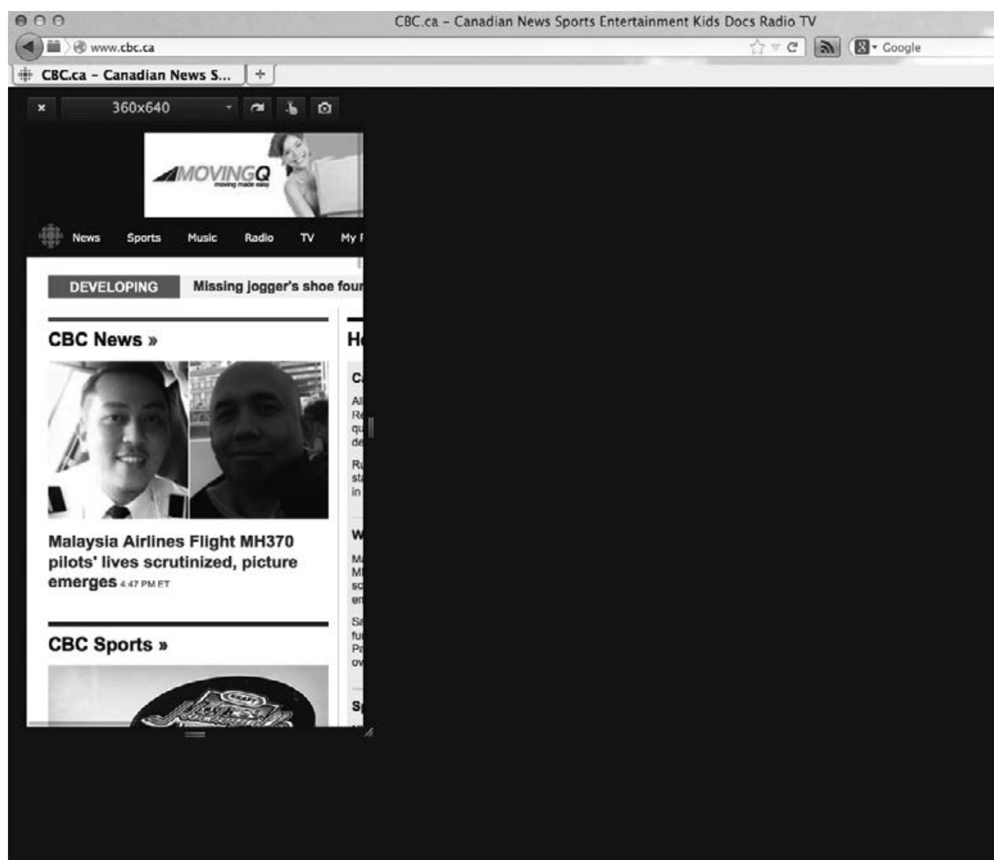


图 8-15：Firefox 中的响应式设计视图（Responsive Design View）

8.11.4　浏览器与操作系统

几年前，我们仅在两个操作系统（Mac OS 和 Microsoft Windows）及少量的浏览器（Firefox、IE、Safari 和 Opera）中测试网站。

然而，现在有数种移动操作系统和许多移动浏览器也需要考虑。下面是主要的移动操作系统及其默认浏览器列表。

- iOS（iPhone、iPad）：Safari
- Android：Chrome（较新的设备）或 Android Browser（较早的设备）
- Windows Phone：Internet Explorer Mobile
- Blackberry：Blackberry Browser

对于每一个操作系统，你应该在其最新版上进行测试，并在条件允许时，也测试一下上一个主要版本，尤其是对 iOS 和 Android。

你可能也希望测试下运行在诺基亚设备上的 Symbian OS，直到 2010 年，它还是全球最受欢迎的操作系统，许多塞班手机仍在使用中。此外还有几个开源移动操作系统可供使用，包括 Firefox OS、Sailfish OS 和 Ubuntu Touch OS。

Kindle Fire 运行 Fire OS 操作系统，其是一个定制版的 Android。

每台手机都带有默认的浏览器，比如刚刚列出的那些，并且很多用户甚至没有意识到他们可以选择在自己的手机上安装其他浏览器，就像在桌面电脑上。除了已经列出的默认浏览器，你还可测试：

- Dolphin Browser（Android、iOS）
- Chrome（Android、iOS）
- Firefox for Mobile（Android）
- Opera Mini（Android、iOS、Blackberry 及其他设备类型）
- Opera Mobile（Android 及其他设备类型）

也不要忘了台式电脑。在 Microsoft Windows 和 Mac OS X 系统上进行测试时，务必在下列主流浏览器上进行测试：

- Chrome（Mac OS X、Microsoft Windows）
- Firefox（Mac OS X、Microsoft Windows）
- Internet Explorer（Microsoft Windows)
- Opera（Mac OS X、Microsoft Windows）
- Safari（Mac OS X、Microsoft Windows）

8.11.5 仿真器与模拟器

实际上如果没有某个特定的设备，通过使用仿真器或模拟器你仍然可以在该设备上进行测试，这种方式将使你能够使用桌面显示器来查看网站在特定的设备上将会是个什么样子。

仿真器与模拟器的区别是，前者会模仿设备的表现，而后者只是显示设备的屏幕显示效果。

很多移动操作系统和厂商都提供有仿真器或模拟器下载，你可以查看 Maximiliano Firtman 的文章“Mobile Emulators & Simulators: The Ultimate Guide”（http://www.mobilexweb.com/emulators），其中提供有一个长长的查阅列表。

不过，并无必要去下载大量不同的软件，你会发现那些基于浏览器的工具更容易使用，它能让你同时测试很多设备或操作系统。

最流行和最全面的两个工具分别是 BrowserStack（http://www.browserstack.com）和 Cross Browser Testing（http://crossbrowsertesting.com），它们基于用户数或用量按月收费。你可以通过网络搜索到很多其他的类似工具，有些是可以免费使用的。

8.11.6 辅助技术

不要忘记某些用户将使用辅助技术来访问你的网站，比如屏幕阅读器。

要确保这些用户能够访问你的网站，你应该至少在一个屏幕阅读器中对网站进行测试。如果你没有预算购买商业软件，以下有几个免费的选项可用。

Mac OS X 有一个系统内置的屏幕阅读器：Voiceover for OS X （http://www.apple.com/accessibility/osx/voiceover/）。

你也可试下 Fangs Screen Reader Emulator（https://addons.mozilla.org/en-US/firefox/addon/fangs-screen-reader-emulator/），一个 Firefox 附加组件。

在用屏幕阅读器进行测试之前，你可以先查看网站在不显示图像（由替代文本代替）时的样子，来看看网站是否仍能被人看懂。在主流浏览器的开发者工具插件中通常都有这个选项。

8.12 总结

在创建响应式网站时，你需要充分考虑用户及用户可能会使用的设备。虽然响应式设计并不是仅为移动设备设计，但把注意力更多地放在移动设备上是很自然的，因为我们对它还不太熟悉，有太多的新功能和局限因素要考虑。只是也不要忘了确保你的设计在桌面显示器屏幕上也有良好的效果。

你的设计不应该只针对特定设备，而应该能在所有设备上都有效地工作。通过把注意力集中在移动优先上，更容易确保网站在移动设备上能有效地工作。

响应式设计不是一个非此即彼的方法。如果你没有资源使网站是完全响应式的，至少可以使其是部分响应式的，这也要好过完全不具响应性的网站。

有许多不同类型的设备可以上网，从手机、平板电脑到台式电脑，以及其他的诸如电子书阅读器、智能手表等设备。触摸是当下我们为移动设备进行设计时最大的一个变数，务必使你的触摸目标足够大，且 JavaScript 事件能正确工作于触摸屏。

你还需要确保网站是人人都可访问（无障碍）的。对此要考虑大量细节，例如如果网站是通过颜色来传达信息的，请确保色盲用户不会错过任何重要的信息。

决定网站支持哪些设备取决于你的资源。最好是用实际设备进行测试，如果做不到，可以用仿真器或模拟器。

在考虑过用户和用户使用的设备之后，我们将跳回到设计中。在下一章中，我们将讲述在响应式网站中如何进行文字排版。

第四部分

设计响应式网站

第 9 章

文字排版

网站意味着交流，向人们传播信息。虽然有些信息可用图像传达，但大多数信息还是通过文字来传达的。

网页上的文字对于信息的有效传播是如此重要，因此请务必确保文字易于阅读且具有视觉吸引力。

我们的目标不仅仅是使文字样式好看，符合网站的整体设计风格，更要为阅读文字的用户创建尽可能好的用户体验。

在设计网站时，我们往往把网页上的文字看作是静态的，就像杂志书页上的文字一样，会对其视觉外观进行过多地控制，试图精确控制文本与页面上其他内容项的排列方式，以及字与字相互之间的视觉效果。

在某种程度上，我们这么做无可厚非。但在创建响应式网站时，我们不应再对文本进行过度的精确控制（精确往往意味着缺乏灵活性），而是应该在设置文本样式时使之无论设计和布局怎样变化都能有良好的显示效果，甚至文本内容不是显示在我们设计的网页之上。

在本章中，我们将首先学习向网站添加字体的相关基本知识，接着将讲述如何设置文本的字体大小，以及什么样的度量单位才最适合响应式网站，然后分析行长（一行文本所容纳的字数）是如何影响可读性的，并向示例站点添加媒体查询来演示不管视口宽度是多少都能将行长保持在推荐的范围内。最后，我们将对文字排版进行更深入的讨论，比如留白以及根据屏幕尺寸的变化你应在何时对字型进行调整。

9.1 始于HTML

当你在考虑文字在屏幕上的显示效果时，你可能想到的第一件事就是用 CSS 设置文字的字体大小、字型、颜色等。

但事实上，对文字施加样式需要从 HTML 开始。记住，阅读你内容的用户所看到的文字可能并没有应用你网站的 CSS。它们可能是在一个无法正确显示或根本不能显示 CSS 的设备上，比如可能使用的是会除去原所有样式的 RSS 阅读器或类似 Instapaper 的服务，或者用户是通过屏幕阅读器的大声朗读来听文本的。

正如我们在本书的前面章节所讲的，在 CSS 对用户不可用的情况下，文本需要用正确的 HTML 元素标记。如果 CSS 不可用，浏览器将对 `<h1>` 和 `<strong>` 这样的 HTML 元素使用默认样式，以确保文字对用户仍然是可读和可用的。

9.2 字型

字型（typeface）是一个网站最重要的视觉元素之一。字型会改变信息的基调，这反过来又会影响信息的解读。

此外，选择的字型是否合适也会影响到网站内容的易读性。错误的字型可能导致信息的传达变得困难。

网站可以使用多种字型。只要它们能和谐共处，不对信息造成视觉干扰，那么使用多种字型并无不妥。

一般来说，应为正文（即段落文本等）选择一个简洁易辨读的字型，目的是使正文文字尽可能地容易阅读。

在对正文字型进行选择时，应在不同的设备上以你计划使用的字体大小来测试文本是否容易阅读，并且要以大段的文本来进行测试，而不只是几句话而已。

对于标题，你有更大的选择余地。因为标题文本通常比较大，字型能够表现出更多的细节和个性。

测试标题字型时，应使用可能会实际用在网站上的标题文本来测试。因为即使字体大小相同，文本的宽度也会随字型的不同而不同，所以要考虑选择的字型是否会在页面上占用太多的水平空间。较窄的字型可避免标题因太宽而自动换行。

要想选出屏幕显示效果良好并能彼此和谐共处的字型有很多要考虑的因素，在本书中不可能一一展开来讲。欲对此有更深入的了解，可以参考以下资料：

- Jason Santa Maria 发表于 *A List Apart* 上的文章“On Web Typography”（http://alistapart.com/article/on-web-typography）；
- Richard Rutter 的 文 章“The Elements of Typographic Style Applied to the Web”（http://webtypography.net/toc/）。

> **[小贴士]**
>
> 虽然单词 font（字体）和 typeface（字型）通常可以互换使用，但它们的含义实际上是不同的。字型是一套设计好的字母、数字和符号，比如 Helvetica 或 Times New Roman 字型。
>
> 而字体实际上指的是保存在你的电脑或网站上的用于在屏幕上生成文字的电子文件，你可以认为它们的区别就像是一首歌和一个 MP3 文件的区别。

字型选择为先

在本章的后面部分，我们将学习如何设置文本的不同 CSS 属性，如 `font-size` 和 `line-length`，使其更易于阅读。

但在开始学习这些属性之前，首先需要决定网站使用的字型。虽然我们可以设置文字的字体大小，但不同的字型在字体大小相同的情况下在屏幕上所占用的空间是不同的。

例如，在图 9-1 中显示的是文本相同，字体大小相同，但字型不同的 10 行文本。因为所选字型不同，每个字符的实际大小略有不同，行的长度也因此而不同。如果设计的网站已将行长控制在最佳范围内，之后又改变了字型，那么行长会随之改变，你将不得不从头开始设计！

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam ac nunc lacus.
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam ac nunc lacus.
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam ac nunc lacus.
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam ac nunc lacus.
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam ac nunc lacus.
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam ac nunc lacus.
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam ac nunc lacus.
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam ac nunc lacus.
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam ac nunc lacus.
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam ac nunc lacus.

图 9-1：不同的字型在字体大小相同的情况下所占据的水平空间是不同的

有成千上万种字体可供我们用在自己的网站中，在本章的后面部分，我们将讲述在哪里可以找到免费或收费的字体，以及如何在网站中使用它们。

9.3　使用字体

过去，设计师在网站中使用的字体只能是已经安装在用户电脑里的字体。但除了极少数存在于所有操作系统中字体，我们没办法知道用户还安装有哪些字体，所以你的选择非常有限。

CSS3 的发布使我们能够将字体嵌进网站中，这意味着网站可以链接一个文件，其包含了显示给定字体所需的全部信息。因此当用户查看网站时，他所用的设备将能够显示你所链接的字体（与安装在用户设备上的默认字体一样）。

字体文件可以与其他网站资源一道托管在你自己的服务器上（记住，字体是数字文件），或者可以使用字体服务网站，其允许你链接它服务器上的字体文件。

线上字体（Web font）在各种设备上的显示效果会有所不同，所以你应如往常一样，在不同的设备上对其进行必要的测试。

9.3.1　设计良好的字体

现在，一切都电子化（数字化）了，很容易获得成千上万可用在网站上的字体。甚至有大量的免费字体，但需要注意的是，免费字体的质量良莠不齐，正所谓一分钱一分货。

字型是由字型设计师（type designer）设计的。设计字型是一项艰苦繁琐的工作，不是说只要简单地把每个字母（笔画）的样式设计出来就行了。

实际上单个字母的设计并不是最困难的部分。棘手之处在于如何使单个字母在文本块中能很好地组合在一起，字母与字母之间的空间（字间距）应根据不同的字母组合而有不同的设定。

如果使用的是制作粗糙的字体，在字母组合成单词和句子时你会注意到其与设计良好字体之间的差异。其字间距看上去是不正确的。

在购买字体时，要先在你的屏幕上观察文本块样本的显示效果。只是浏览字符从 A 至 Z 单独排列时的显示效果来确定字体是否适用并不可靠。

9.3.2　自托管字体

托管自己的线上字体并不困难。

首先我们要获得需要的字体文件，可以是购买收费字体，也可以使用以下网站来搜索免费字体（主要是看它们的线上字体部分，因为它们可能也提供用在电脑上的字体下载，而那并不是我们需要的）。

- Font Squirrel （http://www.fontsquirrel.com）
  所有字体都免费，包括商业用途。

- MyFonts（http://www.myfonts.com/info/webfonts/）
  提供超过 60,000 种字体，基于页面访问量收费。

- Fonts.com（http://www.fonts.com/web-fonts/self-hosting）
  基于页面访问量收费。

下载的字体文件通常是一个 ZIP 压缩文件。压缩包中一般包括数种格式的字体文件，通常也会有一个内含使用许可的文本文件。

将获得的字体文件上传到我们的网站中即完成了自托管的工作。在本节后面的 9.3.4 节中，你将学习如何使用 CSS 将字体样式应用于网站的文本。

9.3.3 字体服务

使用字体服务通常比自己托管字体更容易。虽然有一些免费的字体服务，但通常人们会按月或按年支付字体使用费用。收费方式多样，可以根据使用的字体（字型）数量，或使用字体服务的公司帐号下的用户数，或网站在一段时间内下载字体所耗费的网络流量。

字体服务会提供一个在线接口，可根据各种选项和设置来选择你网站需要的字体，选好之后会得到需要加到你的网站中的 CSS 与（或）JavaScript 代码。在加入代码后，只需在要使用该字体的样式中将字体名称添加到 `font-family` 声明中即可。

下面列出的是当前流行的一些字体服务，你可在限定的页面浏览量或时间段内免费在自己的网站上对字体进行测试。除了免费的 Google Web Fonts，其他网站通常是基于页面浏览量和使用字体服务的网站数目收费的，最低只要每年几美元。

- Google Web Fonts（http://www.google.com/webfonts）
  免费，600 多个字体系列可用。所有字体开源且不限制用途，包括商业项目。

- Font Deck（http://fontdeck.com）
  按每种字体收取年费。

- Fonts.com（http://www.fonts.com/web-fonts）
  两万种字体可用，多种收费方案。

- Typekit（https://typekit.com）
  数千种字体可用，多种收费方案。

- Web Ink（http://www.webink.com）
  数千种字体可用，多种收费方案。

- Webtype（http://www.webtype.com）
  按每种字体收取年费。

在测试字体时还可使用以下在线服务。

- Typecast（http://typecast.com）
  其允许你在浏览器中轻松地测试和比较字体，按月收费，可免费试用 14 天。

9.3.4　链接字体文件

无论是自托管字体文件，还是使用字体服务，都需要让 CSS 知道有哪些字体可在网站中使用。为此，需要使用 `@font-face` 属性声明字体。这是一个要首先完成的单独步骤，这样你才可以将这些字体用于设置网站元素的样式。

如果使用的是字体服务，它会提供使用服务的代码，通常与下面我们所使用的代码类似。

如果是下载字体文件自己托管，也可能会提供有使用代码，你只需要改变 URL 中的目录名即可。将字体文件上传到网站上选定的目录中，这样它们就能被链接使用了。

你的代码应该类似于下面的代码。将此代码添加到你的第一个样式表的最前面（或靠近最前面），因为在开始用该字体样式化文本之前，需要先将其下载到用户电脑上：

```
@font-face {
    font-family: WebFontName;
    src: url('WebFontName.otf');
}
```

几个关键点说明如下。

- 在 `font-family` 中指定的是将在样式表的其他地方引用的字体名。理论上，我们可以随意命名，但通常应使用其实际字体名。
- 如果 `font-family` 中指定的字体名包含多个单词，则要用单引号包起来（即 `Web Font Name`）。
- 确保链接字体文件的 URL 中的路径与字体文件在网站上的位置一致，例如 URL（`../fonts/WebFontName.otf`）。

前面的代码仅包含一个字体文件，但通常每种字体会链接多个文件，因为不同的操作系统支持的字体文件格式是不同的。所以，实际的代码可能更像这样：

```
@font-face {
    font-family: WebFontName;
```

```
    src: url('WebFontName.eot');
    src: url('WebFontName.eot?#iefix') format('embeddedopentype'),
    url('WebFontName.woff') format('woff'),
    url('WebFontName.ttf') format('truetype'),
    url('WebFontName.svg#webfont') format('svg');
    font-weight: normal;
    font-style: normal;
}
```

对于提供了多种格式的同一字体，浏览器将使用它支持的那种格式。

要想获得浏览器器支持的字体文件格式列表，可查看由 Zoe Mickley Gillenwater 摘录自 *Stunning CSS3: A Project-Based Guide to the Latest in CSS* 的“@font-face File Types Browser Support”一文（http://www.stunningcss3.com/resources/fontface-file-types-browser-support.html）。

9.3.5 创建字体栈

在为 HTML 元素设置字体时，我们将使用 `font-family` 属性。在这个 CSS 声明中应包含多种字体，因为你需要有一种候补字体，以防前面定义的字体由于某种原因不能加载（比如字体文件链接错误或者托管主机临时宕机）。

浏览器将按顺序检查列表，使用第一个可用的字体，所以应以选定的字体开始，后面跟着候补字体。这种字体选择列表俗称*字体栈*（font stack）。

列表中的最后一项应该是一个通用的字体系列，通常是 `serif`（衬线字体系列）或 `sans-serif`（无衬线字体系列），这将引用设备上该字体系列所对应的系统默认字体。这完全是作为一种候补手段，很少被浏览器所使用，设置它只是为了以防万一。

字体名称必须完全匹配 `@font-face` 中声明的字体名。如果字体名由多个单词组成，需包含在单引号中。

在接下来的例子中，我们选择 Helvetica Neue 作为正文字体。如果由于某种原因该字体不可用，浏览器会试图寻找 Helvetica 字体，然后是 Arial 字体（两者都是安装在设备上的常用字体）。如果这些都不可用，浏览器将以其默认的 sans-serif 字体代替：

```
body { font-family:'Helvetica Neue', Helvetica, Arial, sans-serif; }
```

在选择将 `font-family` 应用于哪一个元素时，通常，你会设置一种字体应用于网站的绝大多数元素。所以将此字体应用于你的 `<body>` 元素，然后在需要时对其他元素使用不同的字体。

通常，候补字体应尽量与正选字体类似，这样在万一使用了候补字体的情况下页面的显示效果不会看起来完全不同。应用字体栈中的每种字体来测试你的网站，以确保不管是使用哪种字体，都有好的页面显示效果。例如，如果所选的某个字体的字符较宽，可能会改变

页面上文本元素的最佳行长，或使标题拆分成多行。

创建字体栈时，可以包括的字体有：

- 任意已使用 `font-face` 链接的线上字体，无论是通过字体服务还是自托管于你的网站；
- 任意已安装在用户设备上的字体。参见 Scott Granneman 列出的每种操作系统的“默认字体”（http://www.granneman.com/webdev/coding/css/fonts-and-formatting/default-fonts/）；
- 通过使用 `serif` 和 `sans-serif` 指定的浏览器通用字体系列。

一些老的浏览器不支持使用 `font-face` 链接字体文件，因此需要包括系统字体作为候补选项。

9.4 调整文字大小

即便你从不对文本应用样式，也需要注意文本的大小，并确保它合乎设计要求，易于用户阅读。响应式设计的一个优点是你可以按比例而不是绝对值来设置文本的大小，这样，文字的大小可以与视口的大小相匹配。

9.4.1 忘掉像素

我们可以用像素单位来设置文本大小。在这之前，像素一直都是一个合适的设置文本大小的方式，因为它是一个固定的单位，不管你的电脑所连接的显示器有多大。

但现在像素已变得不再适用了。如今的技术可把像素做得很小，高密度屏幕在给定的空间中可以容纳下多得多的像素点，这使得屏幕的显示效果非常清晰。此外，像素的数目在不同类型的屏幕上也各不相同。

因此，像素不再是一种在所有设备上都一致的度量单位。如果还是用像素做度量单位，你的设计可能在目前的大多数设备上显示效果仍然不错。但以后各种各样的设备会越来越多，将来还会发明出各类新设备，因而我们无法相信像素度量单位仍能继续如期望的那样工作。

如果你想让网站是响应式的，并在所有屏幕尺寸和设备类型上都有良好的显示效果，那么，请舍弃像素并使用相对度量单位。

9.4.2 屏幕距离

另一件要考虑的事是，即使你认为像素在所有设备上都是一致的，但实际上也并非如此，因为设备制造商会考虑到生产的屏幕在使用中大概距用户的脸有多远。你的眼睛离屏幕越近，文字看起来就越大。

我们可以做个试验，举起你的 iPhone（其屏幕宽度是 320 像素），靠近电脑，电脑屏幕上

有一个调整成 320 像素宽的浏览器窗口，你会惊讶地发现，虽然都是 320 像素，但它们的显示面积是不同的。如图 9-2 所示。

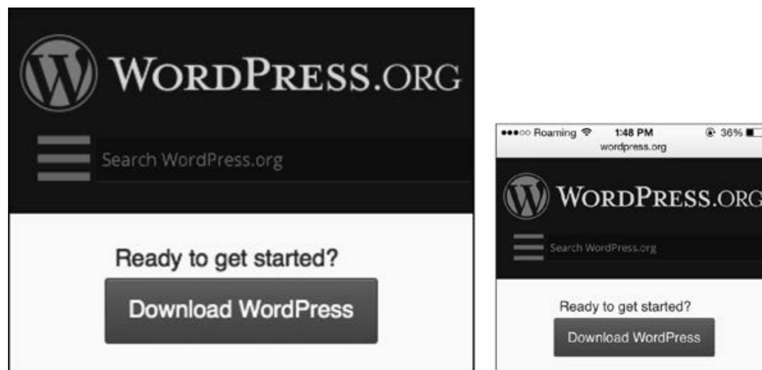


图 9-2：同样的 320 像素宽的网站在不同设备上物理尺寸是不一样的

如果你把两者并排放在一起，电脑上的文字无疑更大，这是必须的，因为电脑显示器通常离你的眼睛更远。

在大多数情况下，你无需对此担心，设备制造商通常都已将此调校好。但是，事物总有出乎意料的时候，所以你需要对各种设备和屏幕尺寸进行测试，来确保不管屏幕尺寸是多大，文本的大小都是合适的、易于阅读的。

阅读 Wilson Minor 的“Relative Readability”（http://wm4.wilsonminer.com/posts/2008/oct/20/relative-readability/）以获得更多有关字体大小和阅读距离的信息。。

9.4.3 绝对与相对

像素是一个绝对度量单位，绝对度量单位拥有一套定义使得其度量值所表示的大小在不同的场景下被认为总是相同的。另一方面，相对度量单位，在定义了一个起始点之后，与其他值之间是彼此相对的关系。

不管是以像素为单位设计网页，亦或是在文字处理软件里设置文档中文本的大小，我们都习惯于使用绝对度量单位。文本都是通过设置的数字度量值来确定其大小的，比如网页中的 14 像素，文字处理软件中的 12 磅，或者是广告牌设计中的 3 英尺。

当使用的是固定大小的画布时（比如纸张），画布的大小是绝不会改变的。但对于一个网站，画布的大小会随屏幕的大小而发生变化，所以需要使网页文本的大小具有灵活性，在必要时可以随同屏幕尺寸一起改变。

因此，响应式设计采用 em 和 rem 这样的相对单位来代替像素那样的绝对单位。相对度量单位简单地说就是其数字度量值所表示的大小是相对于其他事物的大小来说的。我们将很

快对此进行深入的探讨。

如果你曾经从事响应式网站相关的工作，你可能已经知道，可以用一个公式将所有以像素单位表示的字体大小转换成以响应式单位 em 表示。

虽然用公式能很方便地将现有的基于像素的网站转换为响应式网站，但它不应作为一种可持续的处理网站的方法。像素已不再是我们可以倚赖的度量单位，你须要完全停止再考虑以像素为单位来设置文本的大小。

在本书中，我们一直强调在设计中直接使用相对单位，而不是从像素间接转换过来。（如果需要转换现有网站，本章的后面部分将告诉你如何做。）

9.4.4 设置默认字体大小

在设置网站文本的样式时，通常首先要做的是为网站设置默认字体大小。记住，网站中的所有度量单位都应是相对的，全都相对于默认字体大小。

但是以什么值作为设置的起点呢？其实很简单，每个浏览器都能设置自己的默认字体大小，并且每个浏览器会选择一个合适的默认字体大小，大到足以使文本对于大多数用户来说都易于阅读。以此开始设置你的文本，所有你需要做的就是将 `<body>` 元素的字体大小设置为 100%。你可能已经在重置 CSS 时这样做了：

```
body { font-size: 100%; }
```

设置文档的默认字体大小是我们唯一一次使用百分比设置字体大小。其他所有地方都将以 em 或 rem 为单位，所设置的值都是相对于基本字体大小来说的。

如果你好奇 100% 是多大，可以创建一个测试页面来在浏览器中查看它。对于大多数设备，默认字体大小是 16 像素（原谅我刚说了不再考虑使用像素），如果你此刻不想打开浏览器，你可能会在脑海中想 16 像素看上去有多大。

有一些设备会有不同的默认字体大小。例如，Kindle Touch 浏览器的默认字体大小是 26 像素，因为它的屏幕像素密度较高。但 Kindle Touch 上 26 像素的文本看上去类似于其他设备上 16 像素的文本。

字体使用相对大小的优势在于如果要改变网站上所有文本的字体大小，你只需改变基本字体大小即可，所有其他文本的大小会随之改变，但它们仍会维持原来的相对比例。

9.4.5 为什么是100%

尽管在上一节中我们在 `font-size` 中使用 100% 作为 `<body>` 元素的字体大小，但实际上可以使用任何希望的百分比值。然而 100% 却更为合适，正如我前面提到的，浏览器厂商

已经确定了网站上的文本要多大才能使其对于大多数用户来说是容易阅读的，这个大小就是我们设置为 100% 所得到的大小。

不过，在设置网站的默认字体大小为 100% 后，你可能会觉得它看上去太大了。因为你已经看习惯了网站上那些细小的文字。在这之前，很少有网站会把正文文字设置成那么大。但最近几年，许多网站已经开始转向去使用更大的文字。

令人惊讶的是，对此争议一直存在，尤其是在网页设计博客中争论更甚。难以置信会有这么多网页设计师抱怨网站上的文本变得太大了，就像它是对设计师的人格侮辱似的。

的确，对于很多用户来说，文字并不需要那么大。但是，对于另外的很多人来说，大号文本是一个巨大的解脱。如果你视力不好，网页上的小号文本可能很难阅读。据统计，人的视力在 40 岁左右就会减退，从而使得阅读小号文字越来越吃力。我还没 40 岁，但我已经感觉到很多网站的文字太小了，阅读起来有些困难。

实际上在大多数浏览器中可通过快捷键 Command-+（Windows 中是 Ctrl-+）来放大网页上的文字。在文字较小的网站中我会经常使用它，我宁可放大文本来舒服地阅读，也不愿去眯着眼看小字。

一些网页设计师会辩称，因为用户可以自己选择去放大文本，根本就没有必要去设计足够大的易于阅读的文本。但你猜怎么着？不是每个人都知道可以使用 Command-+ 来放大文本。

而且即使用户能这么做，但保证一个网站可用不是*用户*的责任，而应该是*设计师*的责任。

所以应先设置基本字体大小为 100%，以使大多数用户都能轻松阅读页面上的文本。当然，并不是说所有内容文本都要那么大，你可以使用下面将要讲述的相对度量单位使某些元素中的文本（比如页脚）稍微小些。但网站中的正文文本应该至少是浏览器默认字体大小的 100%。

如果你仍觉得 100% 大小的文本看上去太大了，这可能只是因为你的其他设计部分，比如留白或图像，与文字的大小不成比例。就文字而言，如果你无法确定其大小，大文本总是更为合适。

9.4.6 度量单位

文字可使用几种不同的度量单位。

在文字处理软件中编辑文本时你一般是将*磅*（point）用作单位，普通文本可能为 10 磅或 12 磅。1 磅表示 1/72 英寸，因此，12 磅文本就是 1/6 英寸。*派卡*（pica）是一个相关术语，1 派卡就是 12 磅。

在网站中，你可能习惯于使用像素。一个像素理论上是 1/96 英寸。因此，16 像素文本在大多数显示器上是 1/6 英寸大小。

对于响应式网站，我们将使用一个你可能不太熟悉的度量单位：em，其使你能调整字体大小来适配显示器。

1. em

一个 em 等于一个元素当前的字体大小。对于 em 的更详细说明，参考第 4 章中的“Em”。

在下面的代码示例中，<p> 元素的字体大小为 1 em，表示等于页面的默认字体大小，在大多数设备上表现为 16 像素。<h1> 元素的字体大小为 1.5 em，则表示是默认字体大小的 1.5 倍（或 22 像素），如图 9-3 所示：

```
body { font-size: 100%; }
p { font-size: 1em; }
h1 { font-size: 1.5em; }
```

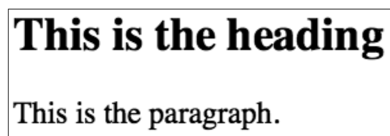


图 9-3：标题文本的字体大小是段落文本的 1.5 倍

如果你现在所工作的网站需要将像素转换成 em，只需将数值除以 16 即可，因此 16 像素等于 1 em，32 像素就等于 2 em。如果嫌计算麻烦，可参考设计师 Jon Tan 制作的“Pixels to Ems Conversion Table for CSS”（http://v1.jontangerine.com/silo/css/pixels-to-ems/），其中列出了假设默认字体大小为 100% 时，各像素大小所对应的 em 值。

2. 嵌套em

在使用 em 作度量单位时可能会让人感到困惑的一点是，它表示的相对大小是基于父元素的字体大小，而不是页面的默认字体大小。例如，假设我们有以下代码：

```
body { font-size: 100%; }
p { font-size: 2em; }
span { font-size: 1.5em; }

<div>This is 1 em.</div>
<p>This is 2 ems <span>(and 1.5 ems)</span>.</p>
```

我们没有对 <div> 元素设置字体大小，所以它将继承页面的默认大小 1 em。

我们设置 <p> 元素的字体大小为 2 em，所以段落中的文本将两倍于 <div> 中的文本。

我们设置 <span> 元素的字体大小是 1.5 em，但它并不是基于默认字体大小，而是基于其包含元素 <p> 的字体大小，而 <p> 元素的字体大小是 2 em。因此，<span> 元素的字体大小是 1.5 乘以 2，也就是 3 em。最终的结果如图 9-4 所示。

This is 1 em.

This is 2 ems (and 1.5 ems).

图 9-4：**<span>** 中的文本太大，因为其 **font-size** 值乘上了包含元素的 **font-size** 值

虽然使用 em 对于嵌套在其他元素中的元素的字体大小可能会产生一些意想不到的结果，但只要在编写 CSS 代码时注意一下，它通常不会是个问题，我们应只对需要的元素设置 `font-size`，而并非所有元素。

3. rem

另一个可用于设置文本字体大小的度量单位是 rem。其与 em 大小相同，只不过 rem 是基于根元素（`<html>`）而不是父元素的字体大小。它比 em 好用，因为不会遇到像 em 那样的嵌套元素的字体大小彼此相乘的问题。

当使用 rem 时，需要在 `<html>` 元素上设置文档的字体大小为 100%，而不是像使用 em 时是在 `<body>` 元素上设置：

```
html { font-size: 100%; }
```

不过，使用 rem 也有一个缺点，那就是并非所有的浏览器都支持它。特别是 IE8 及其更早版本和 Opera Mini，它们完全无法识别使用 rem 为单位设置的字体大小。

解决这个问题有两个办法。

4. 使用回退值

第一种办法，如果想确保这些老式浏览器能按预期的字体大小来显示，那就需要声明一个以像素为单位的*回退值*（fallback value）。

请记住，如果对于一个元素的同一 CSS 属性有两个声明，浏览器将使用后一个。所以我们需要把回退值放在前面，而把 rem 值放在后面。如果浏览器能够识别 rem ，它将使用第二个声明，也就是 rem 值。如果它不能识别 rem，将忽略第二个声明而使用第一个声明，也就是像素值：

```
p { font-size: 18px; font-size: 1.1rem; }
```

要算出与 rem 或 em 等效的以像素为单位的字体大小（基本字体大小是 100%），只需要乘以 16。

[小贴士]

那为什么不直接以 em 为单位设置回退值，这样两个值就可以是相同的了？因为以 em 为单位设置回退值你同样会碰到嵌套元素字体大小会彼此相乘的问题，你需要去确保不会发生冲突。如果你不得不那么处理，那直接用 em 代替 rem 好了。

在所有字体大小声明中都必须包含像素回退值意味着大量额外的工作。还好有另一个更容易的办法。

5. 使用浏览器默认值

第二个办法不是非常精确，但却很简单。如果要支持使用 IE8 及其更早版本的用户你可能只能用这个方法。而且即使你不打算支持这些老式浏览器，仍应该使用这种简便的方法，因为它会对网站的可用性产生很大的影响。

如果你觉得声明回退字体大小太麻烦，你可以让浏览器使用自己为每个元素所设定的默认值（即 < h1> 非常大，<h2> 略小，直至 <p> 使用默认字体大小）。

浏览器会自动实现这一点，除非是像第 4 章所描述的那样添加了一个重置 CSS 到网站中，重新设置了网站上的所有元素都使用默认字体大小的 100%，这将使得它们的大小完全相同。

回到网站的重置 CSS 中，确保 `font-size:100%;` 只应用于 <html> 元素，将其余应用于一连串不同元素的 `font-size:100%;` 样式声明删掉。

这可能导致网站上的字体大小会有些问题，先前设置为 100% 的元素现在则会是一个不同的大小，因为你没有对其以 rem 为单位应用 `font-size` 属性。但如果在开始设置字体大小之前，记得从重置 CSS 中删除不必要的字体大小设置，则不会有这些问题。

正如我提到的，这个选项不是很精确。字体大小不一定是如你所选择的那样。事实上，差不多所有元素都会是基本字体大小的 100%（大到足以阅读），除了标题 <h1> 到 <h6>，它们的字体大小自成一系。在网页中使标题大于正文的字号，而不是与之相同，这样能极大地方便人们阅读。

6. 在em与rem间抉择

那么使用 em 还是 rem 呢？两者各有缺点，如果嵌套的元素设置了 `font-size` 则 em 可能导致怪异的事情发生；而 rem 则需要添加额外的回退代码（除非不打算支持旧浏览器）。

对于大多数样式，em 都将能正确地工作，我们并不会经常遇到嵌套问题。万一遇到了，通常可以对那个设置了字体大小的特定元素进行修改来解决此问题。

使用 em 还有个好处，就是能同时改变网站所有部分的字体大小，比如使 <header> 中的所有文本变得稍微大些。

如果愿意，你也可以在一个网站中同时使用 em 和 rem。

9.4.7　字体大小间的关系

一旦决定了内容从 16 像素的字体大小开始，那接下来该如何设置呢？我们知道标题需要更大些，但多大合适？

许多网站设置 `<h1>` 文本的大小是段落文本大小的两倍，这是一个很好的开始：

```
p { font-size: 1em }
h1 { font-size: 2em }
```

正文文本和标题之间必须要有足够的区别，当用户扫视页面时才能很容易地辨别哪是标题。

除了使用字号来区分标题级别，你还可以使用粗体或不同的颜色来使它们显得更突出。

如果需要使用所有的标题级别（从 `<h1>` 到 `<h6>`），则应该提前规划，使不同的标题级别之间有足够的区别度：

```
p { font-size: 1em }
h1 { font-size: 2em }
h2 { font-size: 1.8em }
h3 { font-size: 1.6em }
h4 { font-size: 1.4em }
h5 { font-size: 1.3em }
h6 { font-size: 1.1em }
```

在大多数情况下我们并不会用到所有的标题级别，因此可以更随意地尝试不同的字体大小来查看其显示效果。

为标题所选择的字体大小会影响到标题的长短。如果标题通常只有几个字，字体大小可以设的更大些以撑满整行。但如果字数较多，则可使字号变小点以尽量避免自动换行，或者是使用较窄的字型。

如果你是个数字控，可以尝试用“黄金比例”来确定标题文字与其他页面元素的大小比。更多的相关信息，请参见 Tim Brown 发表于 *A List Apart* 之上的“More Meaningful Typography”（http://alistapart.com/article/more-meaningful-typography）。

9.4.8　行高

每行文本的高度对易读性而言也很重要。这在印刷术语中称为*行距*（leading），因为它最初是指在印刷机上将铅条插入每排铅字下方来增加行与行之间的空间（leading 作动词可表示加铅条）。

在网页上，所有文本都是呈现在一个矩形盒中（我们在第 3 章中讲过），行高也就是盒子的高度。在视觉上，更大的行高意味着行与行之间有更多的空间。

对于在 line-height 属性中设置的数值应该不带度量单位，以便行高总是与文本成比例。即便是百分比或 em 也不需要，只要指定一个数值即可：

```
p { font-size: 1em; line-height: 1; }
```

line-height 为 1 表示无论字体大小是多少，盒子都与文字一般高（即，从最高的字母的升部的顶端至小写字母降部的底端）。在图 9-5 中第一行文字的 line-height 为 1，而第二行文字的 line-height 为 2，表示行高是字体大小的两倍。你可以看到图中第二行文字多出的垂直空间。

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

图 9-5：**line-height** 分别为 1 和 2 的文本的高度比较

你用于 line-height 的这个无单位数字是与元素的 font-size 相比而得到的，所以行高将根据字体大小自动调整。

文本在由行高创造出的盒子中总是垂直居中，如图 9-5 所示。

对于一个合理的行高设定，其基准值是 1.4，但也可以根据使用的字型或其他因素稍做改变。

如果各行彼此离得太近，用户很难专注于一次只看一行。在图 9-6 中可以看到拥挤的文本难于阅读。

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis ultricies quis dui et auctor. Integer elit justo, lobortis sit amet libero eget, aliquam dapibus nisi. Vivamus quis aliquet enim. In sagittis interdum lorem, ac cursus turpis dictum ac.

图 9-6：**line-height** 为 1 的文本

行与行之间相距太远也不好。如果相距太远，用户的眼睛从一行的结尾移动到下一行的开头会感觉比较累。

设置行高为 2 使得文本行之间的空白过大，不利于舒适地阅读，如图 9-7 所示。这样大间距的行高用在大块的文本上会难以阅读，但可以少量地用在小块文本比如引文上。

在图 9-8 中，我们可以看到，行高为 1.4 使得文本非常适合阅读并考虑到了行与行之间必要的空间。

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis ultricies quis dui et auctor. Integer elit justo, lobortis sit amet libero eget, aliquam dapibus nisi. Vivamus quis aliquet enim. In sagittis interdum lorem, ac cursus turpis dictum ac.

图 9-7：**`line-height`** 为 2 的文本

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis ultricies quis dui et auctor. Integer elit justo, lobortis sit amet libero eget, aliquam dapibus nisi. Vivamus quis aliquet enim. In sagittis interdum lorem, ac cursus turpis dictum ac.

图 9-8：**`line-height`** 为 1.4 的文本

文本的宽度也会影响行高在页面上是否合适。对于更宽的文本行，你需要在两行之间留出的空间也要更多。你可以使用媒体查询来做出这种调整。

另外，笔划粗的重文本相对笔划细的轻文本也需要一个更大的行高。

9.5 行长

从易读性的角度考虑，大块文本的最佳行长（技术术语是行宽）是 45 至 75 个字符，包括空格。

在响应式网站中，你尤其需要注意这一点，因为内容的宽度会随着视口的宽度而变化。

超长的行（如图 9-9 所示）难以阅读，因为你的眼睛会很辛苦地从一行移至下一行。

Wild Animals I Have Known

By Ernest Thompson Seton

These stories are true. Although I have left the strict line of historical truth in many places, the animals in this book were all real characters. They lived the lives I have depicted, and showed the stamp of heroism and personality more strongly by far than it has been in the power of my pen to tell.

I believe that natural history has lost much by the vague general treatment that is so common. What satisfaction would be derived from a ten-page sketch of the habits and customs of Man? How much more profitable it would be to devote that space to the life of some one great man. This is the principle I have endeavored to apply to my animals. The real personality of the individual, and his view of life are my theme, rather than the ways of the race in general, as viewed by a casual and hostile human eye.

This may sound inconsistent in view of my having pieced together some of the characters, but that was made necessary by the fragmentary nature of the records. There is, however, almost no deviation from the truth in Lobo, Bingo, and the Mustang.

Lobo lived his wild romantic life from 1889 to 1894 in the Currumpaw region, as the ranchmen know too well, and died, precisely as related, on January 31, 1894.

图 9-9：太长的文本行难于阅读

太短的行（如图 9-10 所示）阅读起来也累，因为你的眼睛要频繁地来回移动。

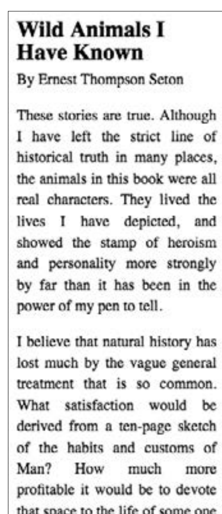


图 9-10：在阅读非常短的文本行时眼睛很快会感到疲劳。

需要说明的是，这些数字只是一个指导方针，并非是一成不变的。你需要在页面上对此进行测试来找出最佳的值。

9.5.1　测试行长

要保证行长总是落在一个可接受的范围内，又不想每次测试都要去数字数，一个简单的方法是将第 45 至第 75 个字符放入一个 `<span>` 元素中，并用 CSS 高亮显示它们，如图 9-11 所示。你可以像本例一样为文本设置一个背景色，或者是将文本颜色设置为红色突出显示在页面上：

```
<p>These stories are true. Although I have left <span
class="testing">the strict line of historical</span>
truth in many places, the animals in this book were all
real characters.</p>

.testing { background-color: #aaa; }
```

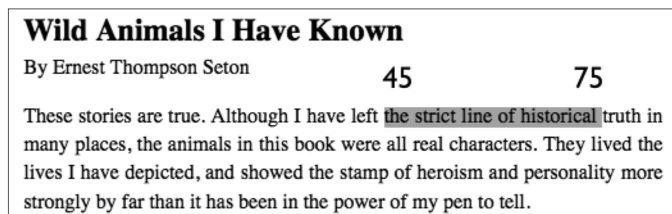


图 9-11：数出第 45 至第 75 个字符并用 CSS 高亮显示它们

如果想对网站做一个快速检查，可以用 Chris Coyier 制作的书签工具做同样的事情（http://codepen.io/chriscoyier/pen/atebf），该工具能将页面上任何文本元素中的第 45 个至第 75 个字符以红色标出。你只需先点击下这个书签工具，然后再点击选择页面上的任一元素，而不需要接触到 HTML 和 CSS。

不过，如果需要频繁地刷新页面来测试自己的网站，你可能会发现添加一个 `<span>` 元素比每次刷新后不得不再次使用书签工具要更方便。

9.5.2　调整外边距及字体大小

你肯定想知道在网站可能是任意宽度的情况下，如何使文本每行的字数总是处在最佳范围内，这其实很简单。

我们将以一个单列布局的网站为例来进行讲解，但你也可以在多列布局的单个列中做同样的事情。

在设计一个网站时，我们会从最小的屏幕大小（手机屏幕）开始，调整浏览器窗口至 320 像素宽。在这个宽度下进行设计，尝试让每行保持在 45~75 个字符这样一个范围内，略微小点也可以。为达成这一目的，你会想使左右两端的外边距尽可能的狭，但又不能为 0，因为这会导致文本挨着屏幕的边缘而不利于阅读。

通过赋予元素 94% 的宽度并设置左 / 右外边距为 `auto`，我们告诉浏览器将剩下的 6% 的宽度平均分给左右两端的外边距，各为 3%：

```
article {
    margin: 15px auto;
    width: 94%;
}
```

接下来，慢慢将浏览器窗口调宽，直到高亮文本完全转到首行的末尾，如图 9-12 所示。这表明在首行有了 75 个字符，如果继续调宽窗口，字数将超过最佳范围的上限，这时候就需要添加一条媒体查询了。

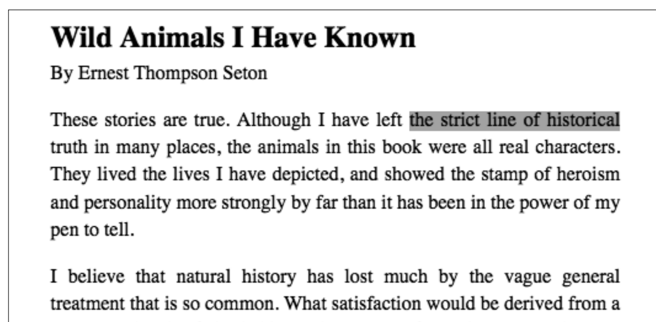


图 9-12：调宽浏览器窗口，直到高亮文本都转到第一行

如果你有一个像 MQTest.io（http://mqtest.io）之类的工具，在同一浏览器窗口的不同标签页中打开，你可以很容易地看到窗口的宽度。在本例中，它是 31 em，所以我们在 31 em 处添加媒体查询。

继续增大浏览器窗口，为使文本行保持在 45~75 个字符的最佳范围之内，我们将增加外边距的宽度（缩小 `<article>` 元素的宽度）：

```
@media screen and (min-width: 31em) {
    article { width: 70% }
}
```

在图 9-13 中，我们会看到，当浏览器窗口比 31 em 略宽时，页面有了更宽的外边距，现在每行文本的字数正好在 45~75 个字符范围的中间。

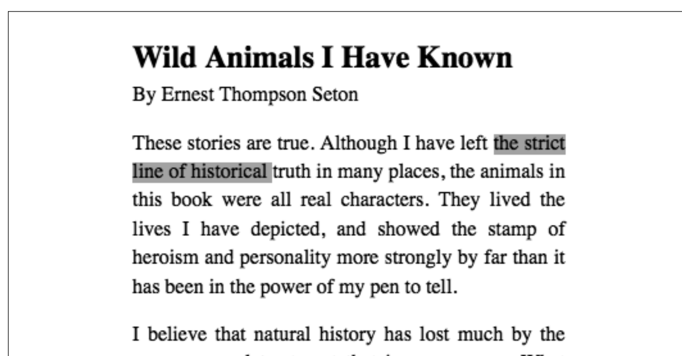


图 9-13：在增加外边距后，第一行的字符数在理想的 45~75 个字符范围中间

继续调宽浏览器窗口，直到高亮文本转到行尾，如图 9-14 所示。

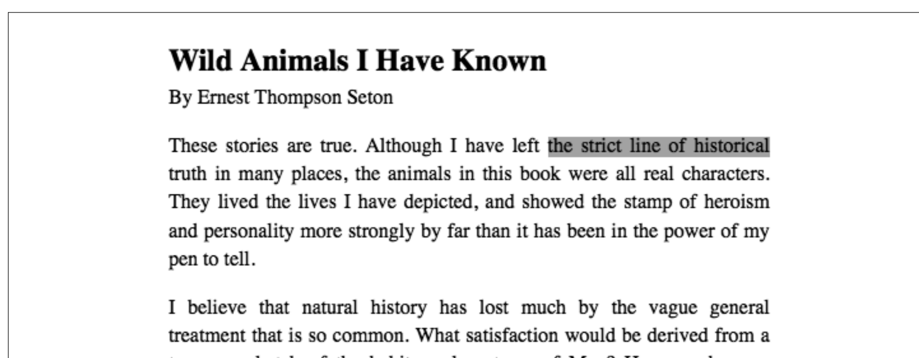


图 9-14：继续调宽浏览器窗口，直到高亮文本全转到第一行末尾

现在我们的视口宽度是 40 em。这一次不再增加外边距，而是将字体大小加大，以减少每行的字数：

```
@media screen and (min-width: 40em) {
    article { font-size: 1.1em }
}
```

结果如图 9-15 所示。

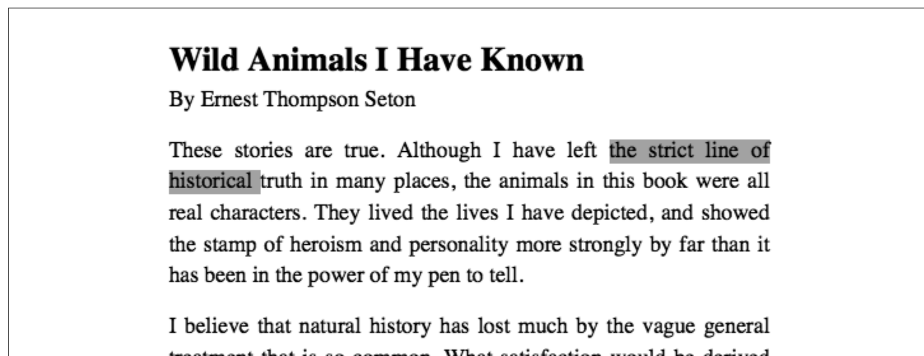


图 9-15：通过增加字体大小，第一行的字符数重新回到了 45~75 个字符的中间范围

你会注意到我们是在 `<article>` 元素上改变字体大小，这将增加元素中所有内容的字体大小，包括标题以及段落文本，因此所有内容的字体大小仍是成比例的。

你可以继续调宽浏览器窗口，增加外边距和（或）字体大小。如果网站有附加的内容，你可以在某个宽度下把它分成多列布局，如我们在第 5 章中学习的那样。

但到了一定的屏幕宽度时，我们并不想让网站继续变宽。

通过增加外边距和字体大小，你可使单列布局在从小屏幕的手机直至标准的桌面显示器上都具有良好的显示效果。但对于更宽的屏幕，总会在某个宽度下，加大文字也不再有意义，到了那个时候，你可以用 `max-width` 限制布局不再变宽，我们在 5.12 节中已有讲述。

9.5.3 断字

断字（也叫作用连字符连接）有益于网页文字排版（断字是指在行尾的单词由于太长而无法完全放下时，浏览器会在适当的位置将该单词分成两部分，并在行尾使用连接符进行连接）。它能让行长变得更加一致，使得阅读变得更容易，设计看起来更完美。断字使得在一行上能容纳更多的字符（如果不使用断字，过长的单词会整体移至下一行），这也意味着更少的滚动。

在网页中，断字属性 `hyphens` 是新的 CSS3 属性。它并不被所有浏览器支持，因此需要使用厂商前缀。`hyphens` 属性可用时能极大地方便文字排版，而不支持它的浏览器也只是不会用连字符连接文本，并无其他不好的影响。下面的代码将使你的网站上的所有文本在换行时会用连字符连接：

```
body {
    -webkit-hyphens: auto;
    -moz-hyphens: auto;
    -ms-hyphens: auto;
    -o-hyphens: auto;
    hyphens: auto;
}
```

有些情况下，你可能不想对某一特定的文本使用断字特性，我们可以排除这些元素。

例如，代码示例中的文本就不应该在换行时用连字符连接，因为在代码中包含连字符往往会让人混淆：

```
code {
    -webkit-hyphens: none;
    -moz-hyphens: none;
    -ms-hyphens: none;
    -o-hyphens: none;
    hyphens: none;
}
```

根据设计，你可能需要关掉某些级别的标题、引用或其他一些元素的断字特性。

请注意，使用断字特性你必须如第 3 章所述的那样设置页面语言：`<html lang=en>`，因为断字特性是与语言相关的。

9.5.4 溢出换行

偶尔我们会遇到很长很长的单词或字符串，在一行文本中容纳不下且不会自动断字。比如我们遇到了一个像 supercalifragilisticexpialidocious 这样长的单词（当然这个词是杜撰的，只是举例方便），或者 URL 中某一很长的部分，其没有能导致自动换行的斜杠或其他字符。

通过添加样式 `overflow-wrap:break-word`，将使得浏览器能对特别长的文字自动换行。

`overflow-wrap` 是 CSS3 中的新属性，还没有浏览器支持它。不过，`overflow-wrap` 只是简单取代了之前存在于 CSS2 中的一个属性——`wordwrap`，因此通过将两者都包含在 CSS 中就可覆盖几乎所有的浏览器：

```
body { word-wrap: break-word; overflow-wrap: breakword; }
```

[小贴士]

顺便说一下，URL 中的下划线字符不会导致自动换行，但连接号会，因此在网站目录名或文件名中最好是使用连接号代替下划线。

9.6 留白

留白作为一种设计理念，指的是布局中留空的部分——外边距，文本行之间的空间，页面上元素四周的空间，以及布局中各列之间的空间。其不一定是白色的，而可以是该区域的背景色。

文本周围需要留白。如果你的文本一直延伸至页面的边缘或紧挨页面上的其他元素，这将会难以阅读。

留白能让眼睛更专注于文本，这增加了易读性。

很多网站都试图一次性地填满尽可能多的信息，这使人很难专注于页面上的特定条目。如图 9-16 所示，这种页面仅是看一眼都会让人感到压力倍增。而通过向行高、内边距以及外边距添加留白，你将创建一个更加均衡和开放的布局。

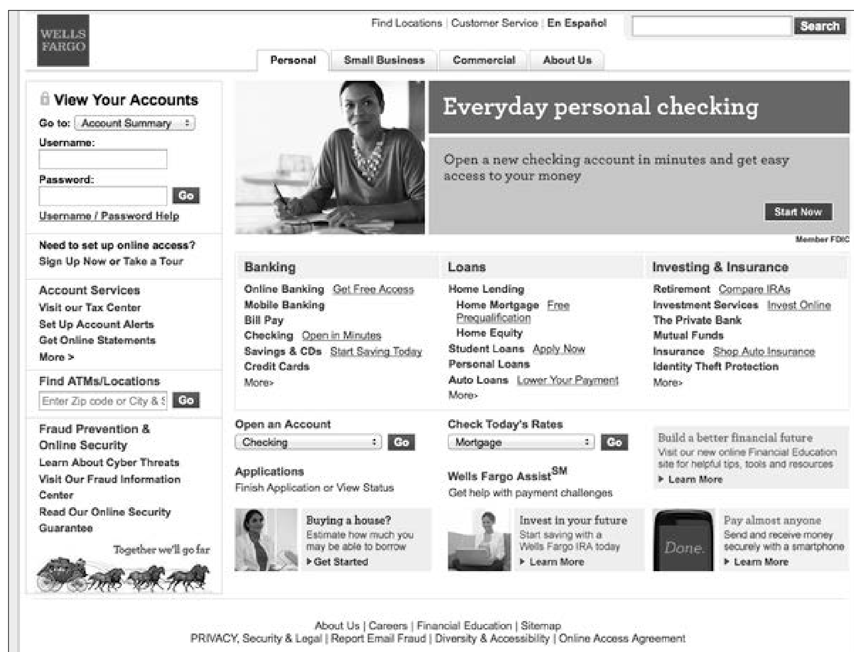


图 9-16：有大量文本而缺少足够留白的网站

想了解更多在网页设计中使用留白的相关信息，可查阅以下文章：

- Paul Boag 发表于 *Boagworld* 之上的“Why whitespace matters”（http://boagworld.com/design/why-whitespace-matters/）；
- Mark Boulton 发表于 *A List Apart* 之上的“Whitespace”（http://alistapart.com/article/whitespace）。

9.7 内边距和外边距

如果你的 CSS 是以重置样式开始的，那么所有内容元素的外边距都会被重置为零，这样你就可以设置自己的外边距值了。首先，你需要设置 <p> 元素和所有标题（<h1> 到 <h6>）的外边距，这样它们彼此之间就有了垂直空间。

这些元素通常并不需要设置左、右外边距（虽然你可能会出于格式上的考虑而加上它们）。顶部和底部外边距应该以 em 单位来设置，以便它们能根据字体大小按比例增减。而内边距与左右外边距不同，它采用百分比单位，因此会根据包含元素的宽度按比例增减。

外边距会根据每个元素的字体大小和字型而变化。较大的字型可能需要一个相对较小的外边距。同样你还不要忘记，行高也能增加元素之间的视觉空间。

我们通常会给标题设置一个较小的 `margin-bottom` 值，那样它会更接近下面的文本，和一个较大的 `margin-top` 值，这样可以把它与前一段文本更好地分隔开来。

一个好的开始是设置所有这些元素的顶部和底部外边距为 1 em，并在设置了 `font-family`、`font-size`、`line-height` 属性之后再单独调整每个元素的边距，直到你对它们的显示效果满意为止：

```
h1, h2, h3, h4, h5, h6, p { margin: 1em 0; }
```

[小贴士]

在这之前，我们讨论过使用 rem 代替 em 来设置字体大小。但在设置外边距时，你应该使用 em，而不是 rem，因为外边距应该与当前元素相对应，而不是根元素。

9.8 为屏幕尺寸改变字型

在较大屏幕上使用装饰性字型能获得很棒的显示效果，因为我们有充足的空间，但要在较小的屏幕上使用这些字型就显得不合时宜了，如图 9-17 所示。

就算图中的简介文字在小屏幕上看起来还算得体，可它占据了整个屏幕，你甚至无法看全整段文字。通常你只需将字体调小些就能解决问题，但对于这个特殊的字型，尤其是它全部使用了大写字母形式，较小的字体大小会难以阅读，所以只能是相对于大屏幕上稍微调小些。

虽然字型是网站品牌的一部分，但必要时我们需要折衷一下。在这个案例中，对于较小的屏幕最好是换一种字型，且不能全部使用大写字母形式。

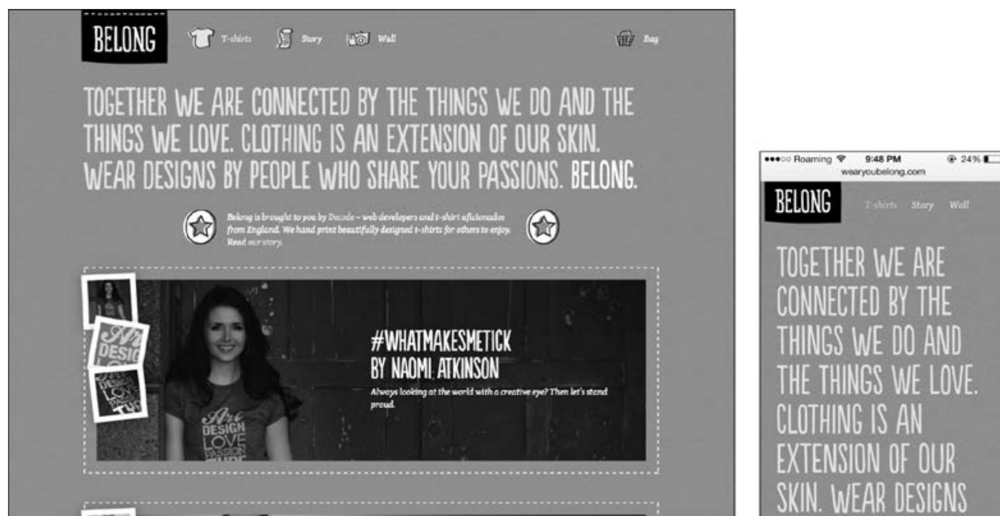


图 9-17：装饰性字型在小屏幕上的显示效果并不好

使用媒体查询能够很容易地完成这个目标。例如，媒体查询可在 30 em 宽度下的狭窄屏幕尺寸上使文字变得较小，将全部大写的文本恢复成正常形式，并把字体系列从当前的 Populaire 变为简单的 Helvetica。整个简介段落将马上变得与屏幕相匹配，也将更容易阅读：

```
#intro { font-size: 1.2em; text-transform: uppercase;
font-family: Populaire, sans-serif; }

@media screen and (max-width:30em) {
    #intro { font-size: 1em; text-transform: none;
    font-family: Helvetica, sans-serif; }
}
```

9.9 总结

网站中的文字是传播信息的关键，因此需要使其更易于阅读并具视觉吸引力。

你首先应该选择好字型。虽然有数千种字型可用，并且很多字型是免费的，但务必选取一种设计良好且适合于你文字的字型。

字体是在屏幕上显示文字所需的数字文件。要想将计划在网站上使用的字体嵌入到网页中有两种方式，自托管字体文件或者是使用字体服务。此外不要忘了指定候补字体，这样当正选字体文件不可得或者不能被浏览器使用时，就可以用候补字体来替补。

在设置网站文本的字体大小时，先以百分比为单位为整个页面设置一个默认的字体大小，然后以 em 或 rem 为单位来设置其他元素的相对大小。正文的字体大小应该至少为 1 em，这样它对于大多数用户来说都是易于阅读的。设置合适的行高对于易读性也有很重要的影响。

恰当的行长会使得文本更易于阅读。使用媒体查询来确保无论页面是在多大的视口中显示，行长都能保持在最佳范围内。留白的合理运用会使页面上的文本看起来更为舒服。

请记住对于响应式网站，你应该放弃对文本字体大小的精确控制。增加灵活性，让你的内容在所有的设备上都有良好的显示效果。

接下来，在第 10 章中，我们将学习如何创建响应式的导航和页眉布局。

第10章

导航及页眉布局

设计响应式网站最具挑战性的任务之一是使网页页眉中的导航以及其他元素成为响应式的。

网页页眉（header）指的是网页顶部包含网站品牌、网站主导航以及用于搜索或站点登录的表单字段等附加组件的那部分区域。网页页眉通常在整个网站中始终都是统一不变的。

页眉中的组件执行两个非常重要的功能：一是告诉用户所访问的是什么网站，二是使用户在网站中导航。

在本章中，我们首先会回到之前的示例站点为其添加一些简单的导航样式。然后我们会观摩一些常见的导航模式实例，看看它们是如何在手机那样小的屏幕上显示导航，以及这些导航布局是如何适合于整个屏幕宽度的。

我们还会讲解如何将网站品牌、搜索框及其他组件整合到一个统一的页眉中。

10.1 响应式导航

不管你的响应式导航的外观如何，成功的关键在于首先从简单直接的 HTML 代码开始，然后再用媒体查询改变导航的 CSS 样式，这样它就能够在不同的视口宽度下都以你希望的方式显示。

以我们的示例站点为样本，我们将添加一些基本的样式使 `<nav>` 元素看起来像是一个真正的导航，然后再添加一条媒体查询在较宽的视口下改变导航样式。

10.1.1 从小屏开始

我们将回到示例网站的最窄视图来开始构建导航。如图 10-1 所示导航是一个无序列表，相关 HTML 代码如下：

```
<nav role="navigation">
<ul>
<li><a href="/">Home</a></li>
<li><a href="/about/">About</a></li>
<li><a href="/links/">Links</a></li>
<li><a href="/contact/">Contact</a></li>
</ul>
</nav>
```



图 10-1：回到示例站点的最窄视图，导航是一个无序列表

10.1.2 样式化列表

我们可以看到无序链接列表可用，但却不是很好看。

因此，为了能有更好的视觉效果，首先我们将添加一些 CSS 样式来去掉列表项符号及内、外边距，使列表项不再缩进（如图 10-2 所示）：

```
nav ul { list-style-type: none; padding: 0; margin: 0; }
nav li { margin: 0; padding: 0; }
```



图 10-2：添加样式去除无序列表的列表项符号和内、外边距

然后，我们将链接置于方框中，这样会更好看些。我们会给每个列表项都加上边框，设置浅背景色，并添加一定的内、外距以使其大小适合触屏操作，如图 10-3 所示：

```
nav ul { list-style-type: none; padding: 0; margin: 25px 0 0; }
nav li { border: 1px solid #666; background-color: #eee;
padding: 10px 1em; margin: 3px 0 0; }
```

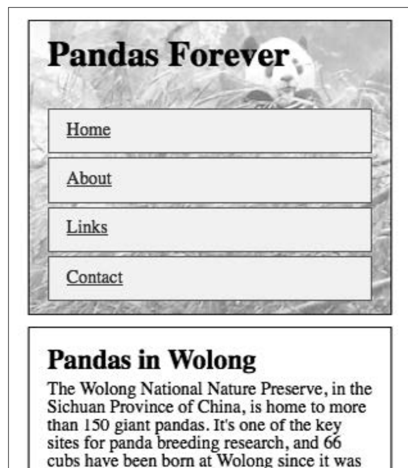


图 10-3：无格式的链接放置于灰底的方框中

接下来，我们要把文字变大一点，使之居中并去掉下划线（如图 10-4 所示）：

```
nav ul { list-style-type: none; padding: 0; margin: 25px 0 0; }
nav li { border: 1px solid #666; background-color: #eee;
padding: 10px 1em; margin: 3px 0 0; text-align: center; }
nav li a { text-decoration: none; font-size: 1.2em; }
```

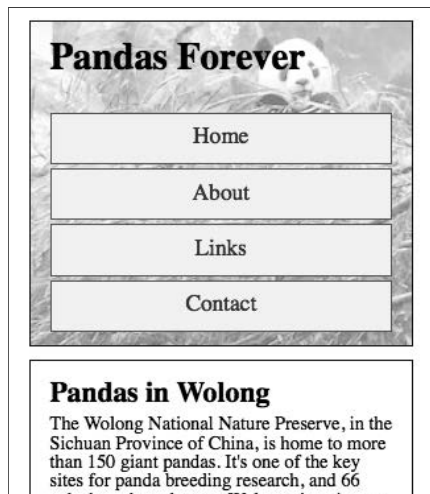


图 10-4：居中文本并去掉下划线使其变得更美观一些

到目前为止，我们主要关心的仍是布局，而不是网站的视觉设计，所以我们只添加了基本的样式。最终的网站肯定要比这好看的多！

但就布局而言，我们现在已经有了一个看起来很棒的适用于最窄视口的导航。接下来我们再来看看导航在更宽的视口中的情况。

10.1.3 水平导航

如我们之前所讲的，我们将从具有最窄视口宽度的浏览器窗口开始，慢慢调整其宽度，当设计开始“变形”（或开始变得不再好看）时，我们将在那个宽度处设置一个断点。

水平导航的断点并不要求与导致整个布局发生改变的断点相同，所以不用去担心如何设置那些断点数值，只需关注导航并找出在什么宽度下它开始变得不再好看。

在视口宽度大约为 30 em（480 像素）时，导航项开始看上去让人觉得太宽了，如图 10-5 所示。

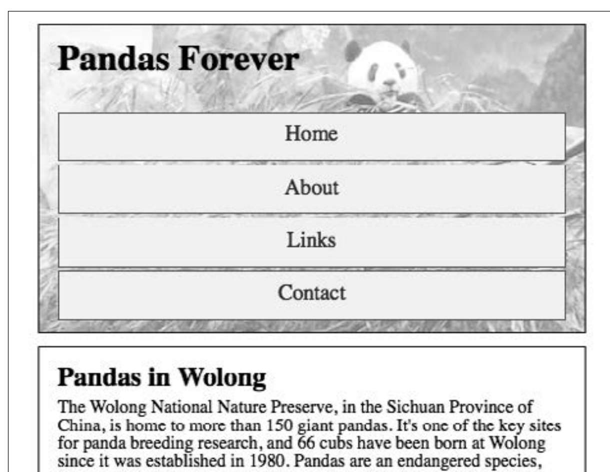


图 10-5：在此视口宽度下，导航链接不再好看

那么在这个点上我们就需要调整导航样式，使所有四个导航项在视口宽度大于 30 em 时排成一行（处在一个水平线上）。

要做到这一点，我们可以给每个列表项添加一个样式 `display: block`，使它们表现像块级元素而不是列表项，然后把它们向左浮动，使它们一个挨着一个地出现在同一行上，如图 10-6 所示：

```
@media only screen and (min-width: 30em) {
    nav li { display: block; float: left; }
}
```

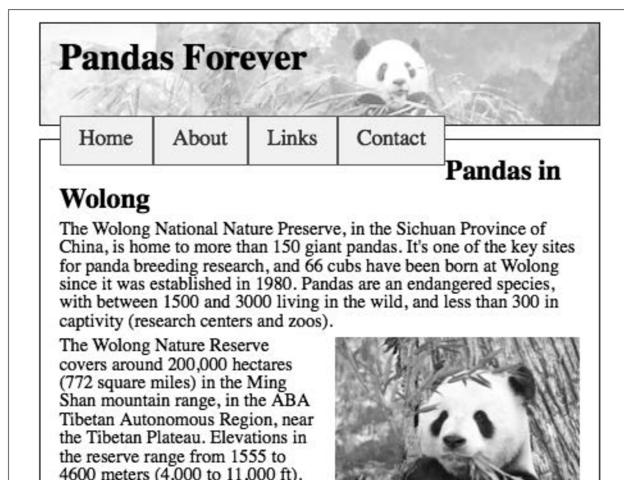


图 10-6：对于更宽的视口，我们将垂直导航变成水平导航，但现在它看上去还不是完全正确

在添加了以上样式之后，出现了两个问题。首先，导航与文章元素的顶部相重叠了。其次，每个导航项的宽度不一（它们的宽度由所包含的文本决定），如果宽度能一致，看起来要更好些。

我们先解决第二个问题。只需设置 `<li>` 元素的宽度为 25% 即可（可在图 10-7 中看到结果）：

```
@media only screen and (min-width: 30em) {
    nav li { display: block; float: left; width: 25%; }
}
```



图 10-7：现在导航项是等宽的了

第一个问题则要费点功夫。通常情况下，我们只要给浮动元素之后的那个元素赋予清除浮动样式（clear）就好了，但是由于 <nav> 是 <header> 元素中的最后一个元素，使用通常的做法并不能使 <header> 元素的底部向下延伸。

解决方法是在 <nav> 元素底部添加内边距来为导航创造出空间。我们将使用 em 作为内边距度量单位，这样内边距能随文字大小的变化而自动调整（如图 10-8 所示）：

```
@media only screen and (min-width: 30em) {
    nav li { display: block; float: left; width: 25%; }
    nav { padding-bottom: 3em; }
}
```

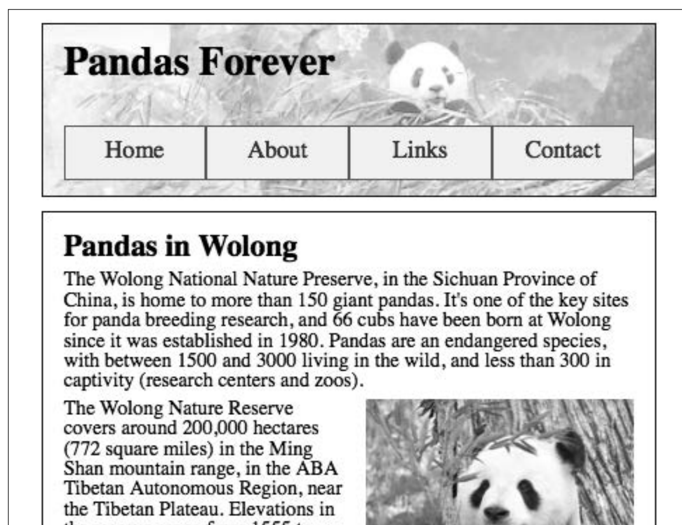


图 10-8：我们修复了导航项错位问题，现在它们不会再跟文章元素重叠了

[小贴士]

不采用添加底部内边距的做法，也可使用以下代码实现相同效果：

```
nav:after { content: ""; display: table; clear: both; }
```

现在的导航样式即使在更宽的视口下看起来仍然合适（如图 10-9 所示），所以不需要再另外添加断点。导航就此搞定！



图 10-9：在更宽的视口下，整个布局看起来仍然不错

10.2　网站品牌

通常，用户访问网站页面时首先看到的是网站品牌。用户从别的地方点击链接过来，他们需要通过品牌来确认自己是否已到达了正确的网站，并且通过站点其他页面导航而来的用户也可通过网站品牌确信他们仍在原来的网站中。

网站品牌不需包含很多内容。通常就是网站或公司标志，甚至就是网站的标题。一般我们还会包含品牌宣传语，这样新的访客来到站点时能确切知道你的公司是做什么的。

对于一个响应式站点，通常只需改变一下网站标志或标题文本大小就能很好地与屏幕适配，但你完全可以做得更有创意些。

在图 10-10 中，设计师并没有在小屏幕设计中使用全尺寸的网站标志，而是对其进行了拆分，将“Dorigati”文本移到了图形的右边，这样处理后所占据的垂直空间要小于全尺寸的网站标志。公司宣传语“Fine wines since 1858”在小屏幕和中等屏幕设计中位于页面顶部，但在宽屏幕设计中其位置则要更下一些，但仍然在首屏上（无需滚动）。

当我们在使用既有的品牌标志（要求总是以相同的排列方式来显示）进行设计时，要确保网站品牌能很好地适配于不同的视口宽度是一个不小的挑战。知名品牌往往对其标志应如何显示有着非常具体明确的风格指南。你应使品牌主认同响应式设计的理念，那样才有希望使他们接受在响应式网站中对公司品牌标志进行灵活处理的手法。

将一堆重要的东西填充到一个狭窄的空间中无疑是一个巨大的挑战。只有使每个组件都更加灵活，才能更好地应对这种挑战。

另一种选择是在宽屏版本的设计中使用公司标志，而在窄屏版本的设计中只使用公司名（文本），如图 10-11 所示。如果在窄屏版设计中不使用公司标志，则仍需要确保其他设计

元素的一致性，这样跨平台访问的用户才能感觉到访问的是同一个网站。在本例中，统一的配色使得品牌在所有屏幕宽度下都有一致的辨识度。



图 10-10：公司标志和公司名称的大小及位置变化取决于屏幕宽度



图 10-11：在 Sprungmarker 网站上，窄屏布局只显示公司名称，而不显示公司标志

10.3 导航链接

导航是网站最重要的部分之一，如果设计得不好，将使得用户很难在整个网站的不同部分间进行导航。

10.3.1 灵活性

在考虑导航在网站设计中应该是什么样子之前，你首先需要决定导航中应包含哪些链接。不要试图在设计中对导航项使用诸如“1 号导航项”这类泛泛的文本，此类文本肯定会与实际导航项中的文本在长度上有很大不同，最终导致看上去完美适配的设计在实际中变得一团糟。

与此同时，即使知道导航项会是哪些，也要明白它们并不是一成不变的。网站或机构的需求可能会改变，将来可能会需要添加或删除链接。你设计的导航是否具有足够的灵活性来应对各种变化呢？

如果最终要将完工的网站交付给客户或团队，这一点显得尤为重要，因为他们可能并没有足够的网页制作技能在后期对设计进行大的改动，那么应确保他们在不破坏设计的情况下能够相对轻松地对导航做些简单的修改。

10.3.2 用户想要做什么

设计响应式导航，如同设计布局一样，应首先从小屏幕开始。

首先为小屏幕设计导航的最大好处在于它迫使你分析目前你的导航中包含了哪些导航项，并选取那些真正重要的导航项。

以往通常是这样来设计网站导航的：1）列出一张包含网站所有组成部分的清单；2）和利益相关者一起对内容链接进行一次卡片分类；3）基于这些类别制作一个多级导航。

但在设计响应式导航时不要这么做。

请记住，首先导航不是给你的利益相关者而是给用户浏览导航用的，应围绕用户如何能够顺利地浏览网站来设计导航。不要把导航变成网站的内容清单，或者更糟糕的情况，把它设计得像公司的组织结构图。请参考 7.1.2 节中的“信息架构”获取有关搭建网站信息架构的更多知识。

在对网站实际所需的内容分门别类，并进行了必要的精减之后。现在我们要考虑的是用户会如何使用这些内容。如果条件满足，你可以使用当前网站的分析数据，看看哪些页面在现实中是用户经常访问的。

英国曼彻斯特市议会网站（http://www.manchester.gov.uk/）是我喜欢的一个响应式网站，如图 10-12 所示。在大多数城市网站上，最突出的内容是现任官员的相关信息与新闻。但曼彻斯特网站不是这样，市议会明显更关注市民的诉求——bin（一个奇妙的英国单词，意思是垃圾回收）肯定不是市政府最迷人的地方，但它绝对是用户经常访问的主题条目之一。

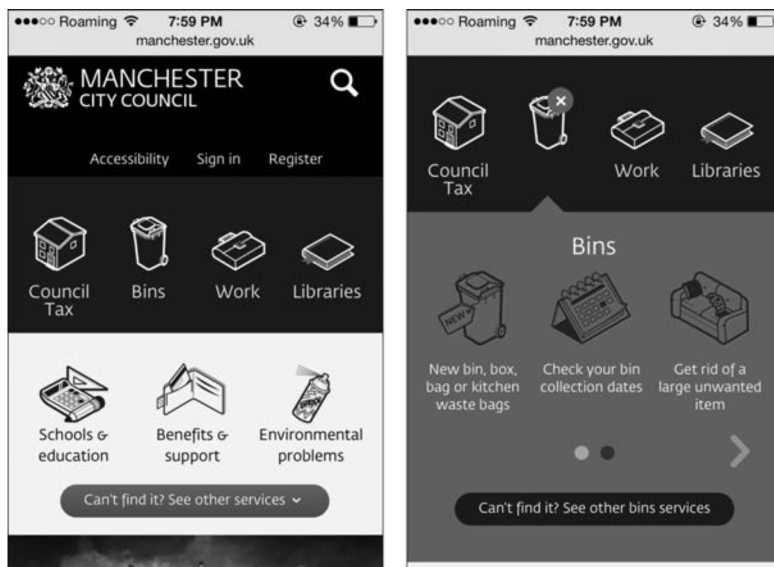


图 10-12：曼彻斯特市议会网站关注用户实际会去搜寻的事物

网站的设计师没有浪费小屏幕上的宝贵空间来刊登市议员的新闻稿或照片，他们关注的是人们访问网站想要获取的信息。

屏幕上被图标所占据的空间如果不用图标的话，可以容纳下更多的选项条目，但由于初始屏幕只是起导航作用，因此使用图标会不显得过于稠密，而且也使得每个触摸目标都易于触按的大小。很多网站都有使用图标，但并没有给这些网站带来实际的好处，而在本案例中，设计师在运用简单明了的图标这方面做得很棒。

此外，网站使用的语言对用户来说也直白易懂，而不是政府部门的“官方”名称。每个人都知道“bin”指的是什么（至少是在英国），这使得它比诸如“Refuse Collection”（废物收集）、“Collection Services”（收集服务），或“Solid Waste & Recycling”（固体废物回收）这样令人费解的短语更容易为用户所理解（这些短语都是我在其他城市网站的导航项中找来的）。

在该网站的宽屏版本中（如图 10-13 所示），设计师有足够空间可以在主要图标的下方添加补充描述文字，并在第二行图标下也多添加了一些人们会频繁访问的链接（例如，“Schools & education”图标目录下包括了“Holiday dates”“Apply for school place”等）。

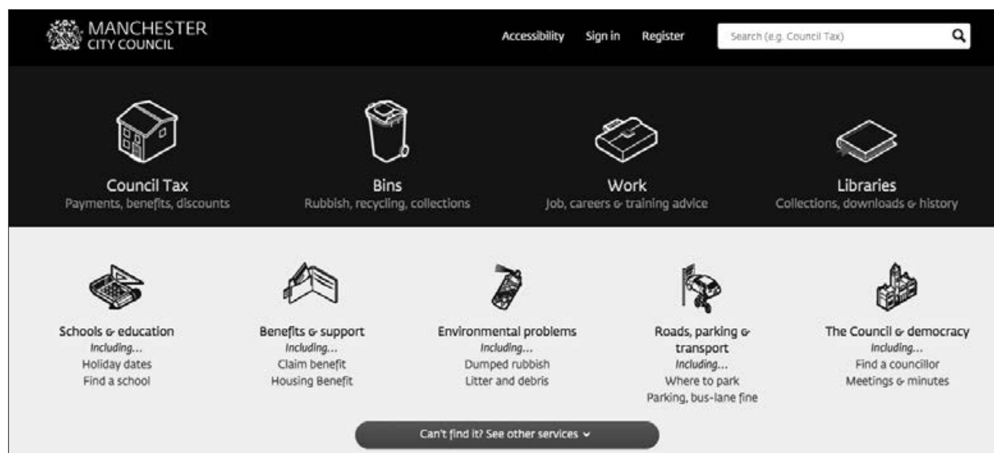


图 10-13：在宽屏布局上，有足够的空间可容下额外的图标及图标下方的补充说明文字

注意，尽管在宽屏设计中可见到更多的图标，但小屏幕用户并没有丢失任何东西。所有的服务链接对他们而言都是可访问的，无论用户使用什么设备，都可通过页面下方的“Can't find it? See other services”（没找到？查看其他服务）按钮来访问所有的服务链接。

10.3.3　基于目标的导航

当为桌面屏幕进行设计时，有足够的空间来添加大量的导航项，很多网站都利用了这一先天优势，毕竟，谁都希望自己的用户能尽可能容易地获得他们所需要的东西。

事实上，在网页设计中有一个存在已久的，非官方的“三次点击”准则，即站点中的每个页面必须不超过三次点击就能到达网站的另一页面。无所不包的多层导航菜单即是遵循此准则的产物。

这在理论上听起来挺不错——点击次数越少越好——但如果用户不知道该去点击哪里，那他们的点击次数其实是无关紧要的。

研究表明，只要用户相信自己正走在通往目标的正确的道路上，只要用户觉得每次点击又向目标迈进了一步，那他们并不会介意额外的点击。

事实上，他们介意的是那些令人沮丧的点击，这让他们感觉似乎被送往错误的方向，或者并没有让他们感觉到接近了目标。

在图 10-14 中，你会看见几年前 GoDaddy 网站的导航。导航中的每个选项卡都包含多个选项，当鼠标悬停在这些选项上时又会弹出对应的子菜单。

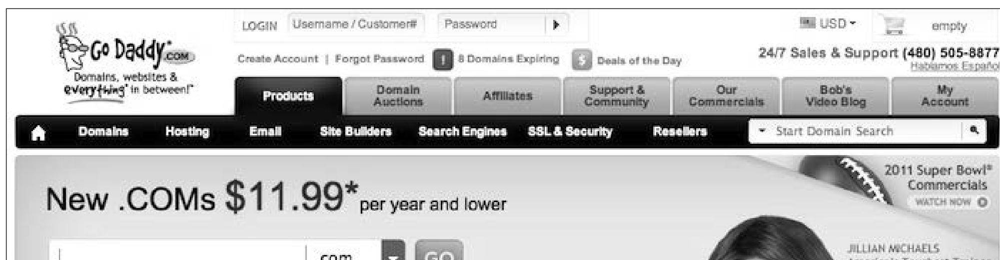


图 10-14：旧版本的 GoDaddy 网站有非常多的选项，新客户可能无法弄清所有选项的意思

这样的导航方式的确只需点击几次就能访问网站的任一部分，但前提是你必须确切地知道自己在寻找什么，并且能够估计出它归于哪一分类。

对有技术背景的人来说，这可能不是一个大问题，但对于没有技术背景的潜在客户，比如想创建他们的第一个网站的小企业主，这样的导航会把他们搞得晕头转向。如果初始过程就太难，客户很可能会放弃，转而去寻找其他无需大费周折就能满足他们需求的网站。

如图 10-15 所示，经过重新设计后，网站的导航变得简单了许多。很明显，GoDaddy 在试图拉拢新的客户。四个最显著的菜单项是基于目标，而不是产品的。对于那些不熟悉网站术语的客户而言，“Build your Website”（建立你的网站）给他们指出了一条非常清晰的路径。而对于那些明白自己想要什么的用户，所有其他的选项依然存在，只不过被放在了“All Products”（所有产品）下面，访问它们只需多点击一次，这一点儿都不麻烦。



图 10-15：新版本的 GoDaddy 网站有清晰的路径来指引新用户

虽然 GoDaddy 不是一个响应式网站，但它的导航却是一个很好的基于目标的导航的例子。GoDaddy 在其单独的手机版网站上的导航也与此类似。

10.3.4 保持一致性

不管你对不同视口宽度下的响应式网站的导航做了何种改变，要牢记对于你的用户，应该让他们感觉网站还是原来的网站。如果用户在手机上看到的导航与他们在桌面屏幕上看到的导航迥然不同，那么他们可能会为此感到困惑不已。

根据所访问设备屏幕宽度的不同，呈现给用户的导航也会不同，但其主要导航项应该一致，并且是以同样的顺序排列。如果在桌面宽度的设计中那些频繁访问的功能链接（比如登录）被突出显示在顶部，那么你的用户很可能也希望在网站的小屏幕版本中找到这些链接。

对于具有单独的手机版站点和桌面版站点的网站而言，导航存在不同是很普遍的，但在响应式网站上也会有导航不同这个问题。

例如，看一下图 10-16 中宜家网站的桌面版和手机版。桌面版网站有一个“Departments”（样板间）列表，手机版网站有一个“Products”（产品）列表。

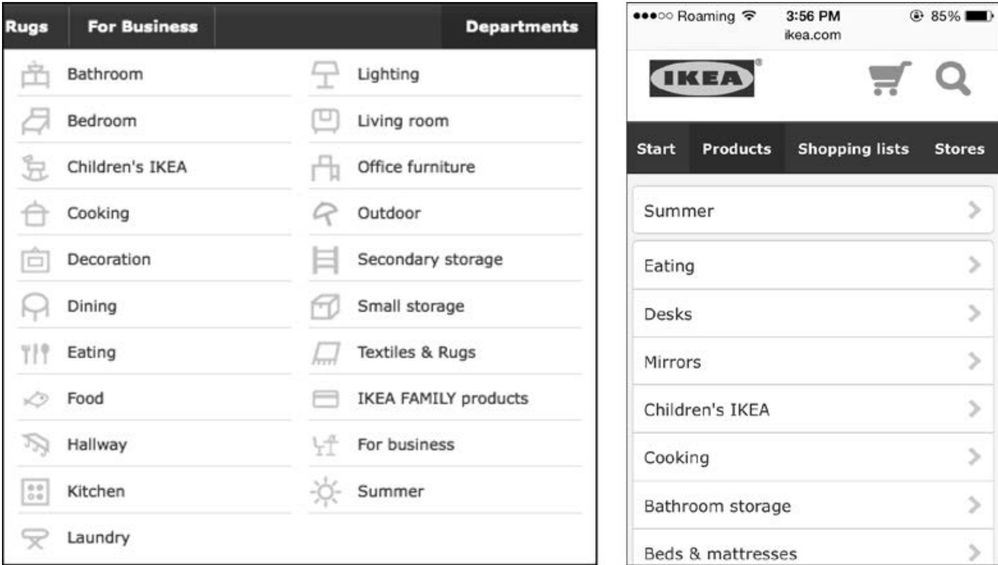


图 10-16：宜家网站桌面版和手机版的导航所包含的类别略有不同

这两个列表很相似。例如，它们都有“Cooking”（烹饪）和“Decoration”（装饰）类别。而桌面版网站有“Dining”（餐饮），手机版网站则有“Eating”（饮食），这两个类别实际指向的是相同的链接。

不过，桌面网站有“Bedroom”（卧室）类别，包括床、床垫、床上用品，以及卧室家具（比如床头柜），而手机网站只有“Beds & mattresses”（床和床垫），尽管有一个单独的床上用品链接，但似乎没有哪个类别选项看上去包含床头柜。

现在，尽管我相信对于哪种导航更好会有很多争论，但真正的问题在于它们是不同的。

举个例子，如果你在午餐时间用办公电脑浏览该网站，挑选了一个床头柜，你记得自己是通过“Bedroom”链接达到床头柜部分的。然后再假设那天晚上你在看电视时，想再看一下床头柜，便拿出了手机浏览网站，可能你没有注意到导航的不同，而是直接点击

“Products”（产品）类别，其包含的选项看起来与午餐时所看到的相似，接着你会点击“Beds & mattresses”（床和床垫），认为它与你在桌面网站所点击的链接是相同的，可结果却是你搞不明白为什么现在找不到床头柜了。

这是拥有单独的桌面版和手机版网站的一个缺点，你很有可能最终会弄出两个不同的用户界面，它们很容易把人搞糊涂。而且这个问题也延续到了响应式网站上。

设计师们常常会假定移动设备用户想要的选项与桌面用户不同，并在响应式网站的小屏幕版设计中隐藏某些选项。

然而，你需要记住的是，移动设备用户和桌面用户往往就是同一个的用户，只不过是使用了不同的设备，他们认为网站的导航是相同的。

10.3.5　保持简单

我们已经知道需要确保响应式网站能工作于所有的设备，不管设备的性能高低和屏幕大小。这对于而言导航尤其重要，因为如果导航对某些用户不起作用，那么网站对他们来说就基本上是不可用的了。

切记，同布局设计一样，为了那些所用设备不支持媒体查询或 JavaScript 的用户，导航也需要从基本的 HTML 开始。

尽管一些更复杂的响应式导航会依赖于 JavaScript，但你需要确保导航在任何屏幕大小和任何设备类型上都可用（即导航项是可见和可点击的）。不支持 JavaScript 的用户可能只占非常小的比例，因此可以不必追求很棒的视觉效果，保证能用即可。

同时，你也应明白越复杂的代码越难以维护，且在对网站进行更改时，也越容易出错。考虑清楚你是否真的需要那么花哨的导航，还是一排简单的链接就够了。其中的权衡是否合适呢？

10.4　导航模式

大多数非响应式网站都是为桌面显示器设计的，它们的主导航遵循着同样的通用模式：以通栏方式水平显示在页面顶部，如图 10-17 所示。

左侧导航在几年前很流行，但随着时间的发展这种导航模式已经过时，正如研究也表明左侧导航远不及顶部导航的效率。

虽然有时在主流网站上还能看到左侧导航（如图 10-18 所示），但它通常是用作主题列表，而主要的功能导航还是在页面顶部。

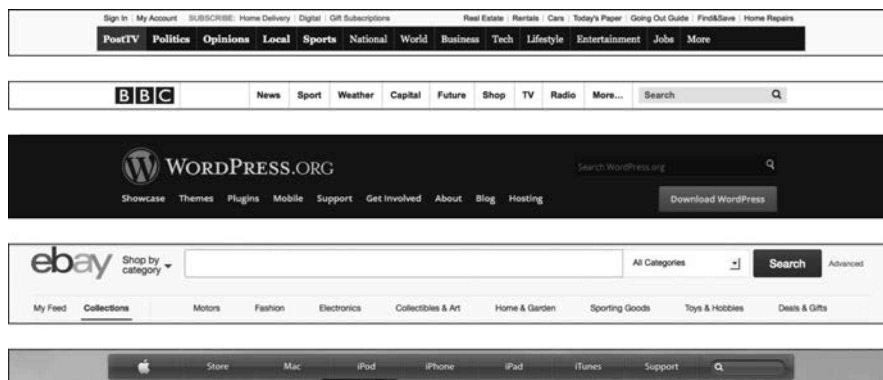


图 10-17：大多数桌面尺寸网站设计在顶部都有一个水平的通栏主导航



图 10-18：左侧导航有时用于主题或类别列表，而主导航项仍在顶部

响应式导航能适应所有的视口宽度，从手机至宽屏显示器。在大多数情况下，响应式导航将遵循现有的宽屏导航设计模式，因为这些模式已为上网的人们所熟知。

在首次设计响应式网站时，许多设计师发现他们所面临的最大的挑战是制作导航，但问题实际上并不是制作一个响应式导航，而是制作一个好的小屏幕导航。而一旦有了好的小屏

幕导航，在较宽的屏幕上添加媒体查询来改变或移动导航不过是小菜一碟。

现在有很多非常棒的小屏幕导航设计，它们通常遵循数个模式中的一个，也就是说，我们可以根据共同的特征对它们进行分类。

我们将讲解一些基本的响应式导航模式，解释它们的工作原理并给出实际的网站用例。

除了在本书中讲到的模式，还有很多其他的模式，Brad Frost 的 Responsive Patterns（响应式模式）网站（http://bradfrost.github.io/this-is-responsive/patterns.html）是一个很好的资源。其中不仅包含了导航模式，还有用 HTML 和 CSS 实现的可供你剖析的例子，以及响应式布局、表单及其他模块的设计模式。

不要忘记，你应该使用 `<nav>` 元素和相应的 WAI-ARIA 的角色属性来编写导航代码，如我们在第 3 章中所讲到的那样：

```
<nav role="navigation">
 ...
</nav>
```

[小贴士]

要想获得更多的移动设备用户界面模式，可查看 Theresa Neil 所著的《移动应用 UI 设计模式》一书（http://www.mobiledesignpatterngallery.com）。

10.4.1 顶部导航

在响应式网站上最简单的导航处理方式就是把所有导航项都放在页面顶部，再根据屏幕宽度用媒体查询和 CSS 布局样式重新排列它们。

在本章前面部分我们用示例站点进行讲解时，你已经见过了简单的顶部导航代码，在这里，我们再来看几个真实的网站例子。

你可能会想把导航项先是在小屏幕上以垂直堆叠方式排列，然后在更宽的视口中，转而以水平并排的方式排列，如图 10-19 所示的 Enochs Fish & Chips 网站那样。



图 10-19：顶部导航中的项目会随视口宽度变化而重新排列

这个例子中的 HTML 代码非常简单：

```
<ul class="nav">
    <li><a href="#food">Food</a></li>
    <li><a href="#fish">Fish</a></li>
    <li><a href="#news">News</a></li>
    <li><a href="#contact">Contact</a></li>
</ul>
```

这之后再用媒体查询来链接单独的样式表，根据屏幕宽度使用绝对定位来改变导航项的位置。

如果你之前听说过响应式网站看起来总显得乏味和沉闷，那么 Enoch 的网站无疑证明了它们并不都是那样的。

再来看第二个实例，在不同的视口宽度下，Food Sense 网站也会对页面顶部的导航项进行重排。在图 10-20 中，你可以看到在不同的视口宽度下导航项和网站 Logo 是如何在页面顶部（或页面左侧）排列的。

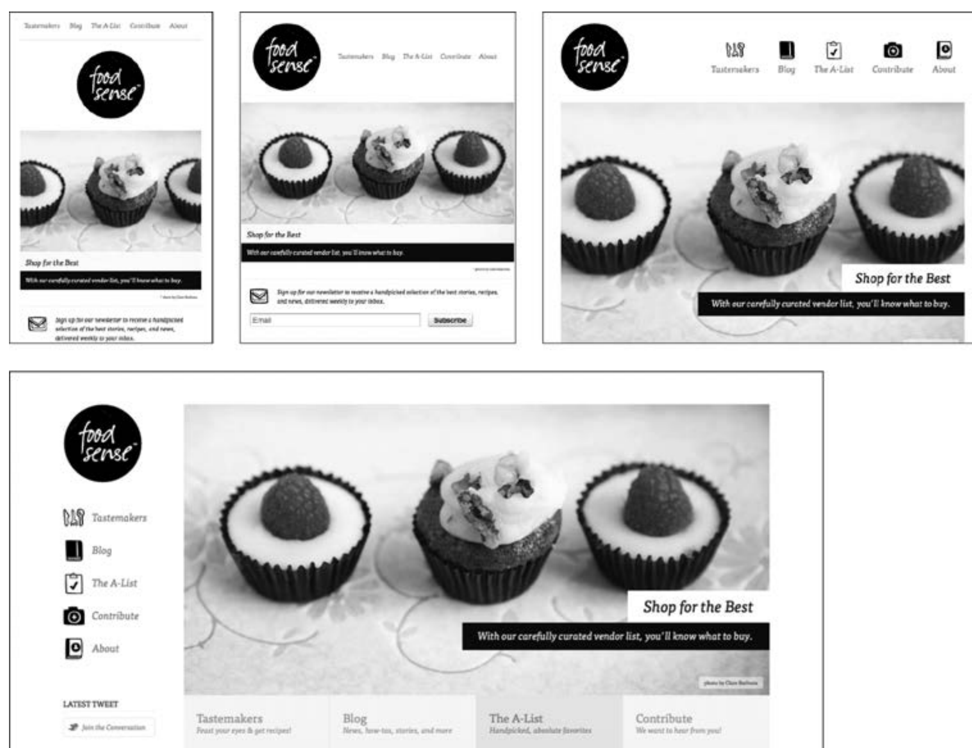


图 10-20：可根据视口宽度，用媒体查询和基本布局样式重新排列页面顶部的导航和网站品牌

该网站页面顶部的元素都包含在 <header> 元素中，导航则是 <nav> 元素中的一个无序列表，网站标志是一个 <h1> 元素。同样的，在不同的视口宽度下网站通过媒体查询用基本的布局样式比如浮动和外边距来重新排列导航项。

在较宽的页面布局中每个导航项旁都有一个图标，这是通过背景图来实现的，宽度较窄时用 `background-image: none` 将其去除。

不过，如果你有非常多的导航项，这样的方法就行不通了。因为在小屏幕上，导航最终将会占据整个手机屏幕，如图 10-21 所示。

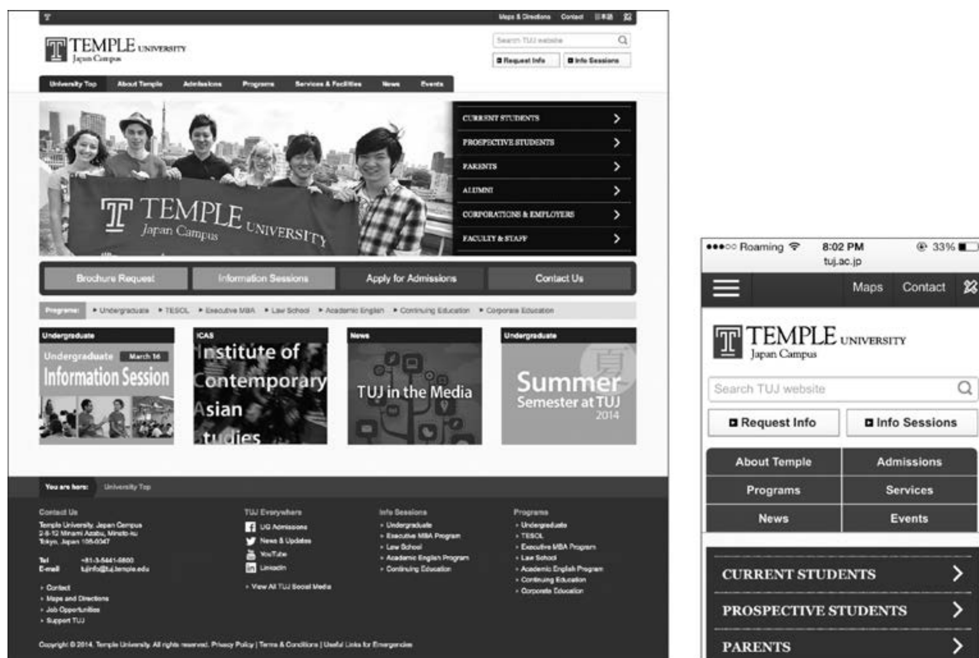


图 10-21：如果你有太多的导航项，在移动设备上它们最终将会占据整个屏幕

你的导航解决方案必须在小屏幕上为内容留出空间。要确保用户访问网站时（即使用的是手机），一眼就能够看到一些有趣的内容，而无需滚动屏幕。

10.4.2 页脚导航

要解决在网站的小屏版本中导航占用空间过多的问题，最简单办法就是将其移到页面的底部，然后再用一个锚点链接（一个可将你带到同一页面的不同位置的链接）指向它，这通常称为*页脚导航*（footer navigation）或*页脚锚点导航*（footer anchor navigation）。

Contents Magazine 网站使用的即是这种类型的导航。在图 10-22 中，你可以看到在页面顶部有一个“Explore”链接，包含一个下指的箭头，点击该链接将把你带到靠近页面底部，包含有搜索框和导航链接的地方。

尽管这个网站只有四个导航链接，但这些链接加上搜索框在小屏幕上几乎占据了一半的可用空间，因此把它们迁移出页面顶部将为网站品牌及开头的页面内容腾出更多的空间。

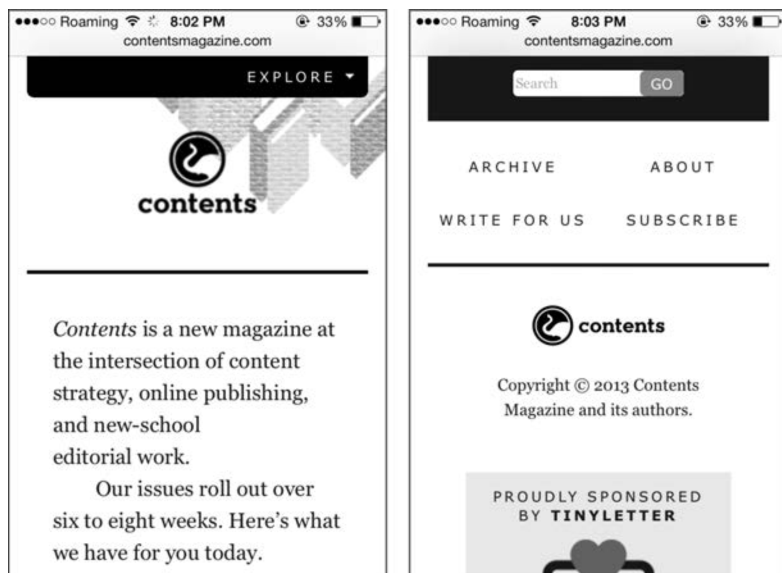


图 10-22：Contents Magazine 网站使用页脚导航

实现页脚导航的代码非常简单，只需要在页面顶部放一个锚点链接，下面的代码复制自 Contents 网站：

```
<p class="go-nav">
    <a href="#site-nav"><b>Explore</b></a>
</p>
```

搜索框和链接的代码放在靠近 HTML 文档底部的位置，与它们在小屏设计中的实际位置相吻合。在本案例中，网站的设计者用一个 `<div>` 元素包含搜索框和导航链接，其中导航链接作为一个 `<ul>` 元素中又包含在 `<nav>` 元素中。

在更大的视口宽度下，顶部有足够的空间来容纳导航及搜索框，如图 10-23 所示。



图 10-23：在更宽的视口下，导航移至屏幕顶部

因此网站使用媒体查询，通过绝对定位技术将包含导航项的整个 `<div>` 元素移到页面顶部：

```
@media screen and (min-width: 48em) {
    #site-nav { position: absolute; top: -5em; width:
    100%; z-index: 5; }
}
```

这时就不再需要“Explore”链接了，因此应将其从屏幕上隐藏。通过赋予其一个负的位置值将其置于视口之外从而达到隐藏的目的：

```
.go-nav { left: -1000em; }
```

如果你只有几个链接，那么页脚导航是一个好的解决方案，但缺点是它可能会迷惑用户——怎么点击一个链接，就突然到了页面的底部。

10.4.3　切换式推出型导航

星巴克网站在较窄的屏幕宽度下使用一种常见的通常称为*切换式菜单*（toggle menu）的导航，该导航可通过反复点击使其在隐藏和可见状态下来回切换。

在图 10-24 最左边的那张图中，你看到的是星巴克网站的窄屏设计。该图右上角的三道横线是一种常见的导航图标，点击它后，导航显露，将其下面的内容往页面下方推。你可以看到，根据屏幕宽度的不同，导航会分成一列或者两列。

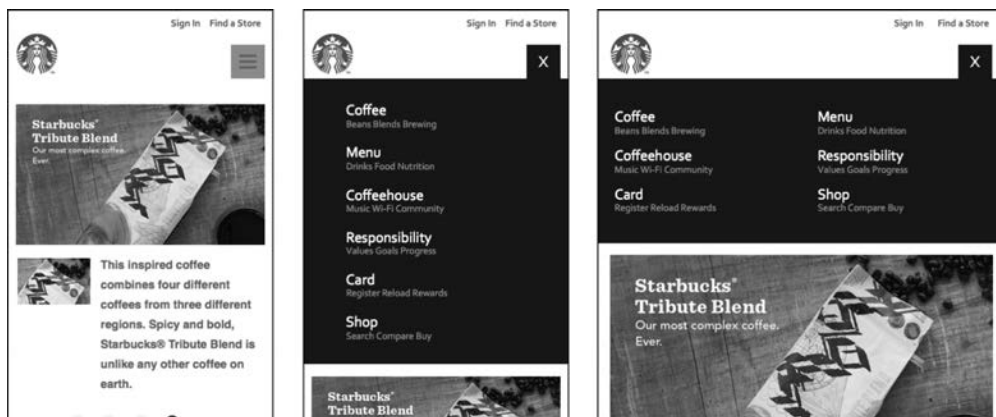


图 10-24：点击屏幕上方的三道线图标使导航显现，这会将页面内容往屏幕下方推

在导航处于可见状态时，只需点一下“×”即可让它消失。

在更宽的屏幕中（如图 10-25 所示），页面顶部有横向空间可容纳下整个导航，附加的媒体查询在有更多空间可用时会进一步重排导航组件。

对此种导航模式需要注意的就是其有些复杂，并且要用到 JavaScript。如果你不懂 JavaScript，那也不用担心，这样的例子比比皆是，你可以直接复制粘贴代码到自己的页面中。

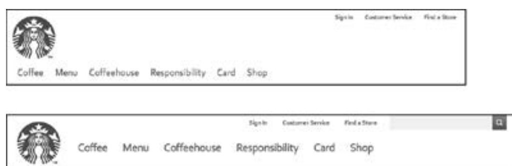


图 10-25：在宽屏幕上浏览时，导航以水平排列方式显现

你可以在 Brad Frost 的响应式模式汇集：切换式导航（http://codepen.io/bradfrost/full/sHvaz）中找到实现这类导航的代码。

在 Frost 给出的例子中（像在图 10-24 中一样），小屏幕版导航的工作方式是，当用户点击图标时，会触发一个 JavaScript 动作，调用 jQuery 的 `toggleClass` 方法将 CSS 类 `active` 添加到 `<nav>` 元素中，下面结合代码详细说明。

在首次加载页面时，`<nav>` 元素不具有 `active` 类属性，应用的是以下样式：

```
nav { max-height: 0; overflow: hidden; }
```

其中 `max-height` 为 0 表示元素在屏幕上不占有空间，`overflow: hidden` 表示 `<nav>` 元素包含的内容不得超出元素范围显示，合在一起的意思就是 `<nav>` 元素是不可见的。

当用户点击图标时，触发 JavaScript 将 `active` 类添加到 `<nav>` 元素中，以下 CSS 样式被应用：

```
nav.active { max-height: 15em; }
```

`<nav>` 元素的 `max-height` 属性变成 15 em，这意味着元素可以在屏幕上获得它所需的全部空间（最大高度 15 em，这比其实际所需高度要大很多）。因此，该元素变成可见状态，其后面的内容会往下推。

导航项是一个无序列表，因此它们以垂直堆叠方式显示（当然可通过附加的样式使之变得更好看）。

对于更宽的屏幕，如图 10-25 所示，将应用以下 CSS 样式：

```
@media screen and (min-width: 48.25em) {
    nav { max-height: none; }
    nav li { display: inline-block; }
    a.menu-link { display: none; }
}
```

其中 `<nav>` 元素的 `max-height` 设置为 none，表示导航将以全高显示。列表项被赋予样式 `display: inline-block`，这样它们将排列成一行。“Menu”链接被赋予了 `display: none` 样式使得它不再可见，因为菜单已经是可见状态，我们不再需要显示切

换菜单的选项。

不要忘记确保你的导航对于禁用了 JavaScript 的用户也是可用的。就像星巴克网站的首页设计虽然依赖于 JavaScript，但在禁用 Javascript 的情况下加载星巴克网站，在窄屏上显示的菜单默认已经是打开了的，且在你访问网站的整个过程中它都保持着打开的状态，这样它对用户仍然是可用的。不过遗憾的是，由于星巴克首页的设计使用了 JavaScript，你会发现禁用 JavaScript 后的显示效果存在一些缺陷，并不是所有的内容都是可见的，如图 10-26 所示。

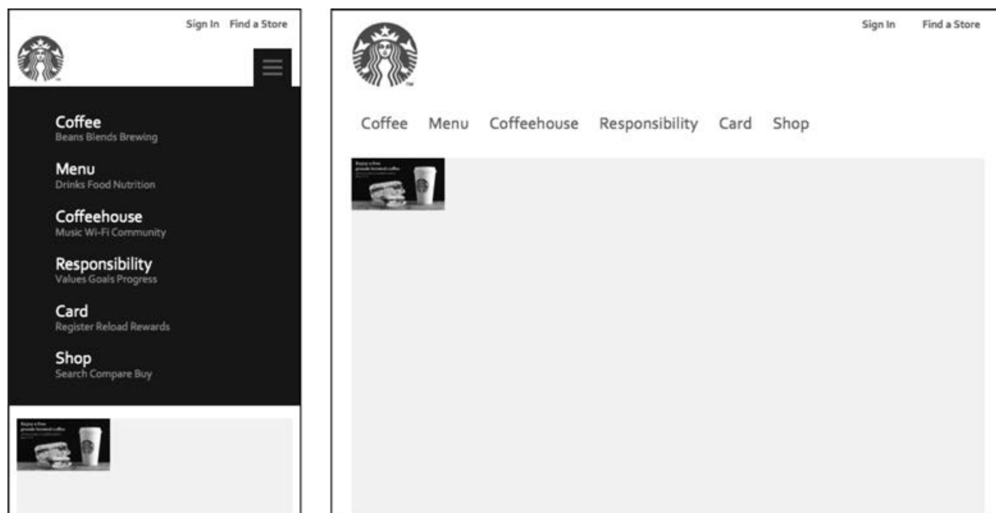


图 10-26：当 JavaScript 被禁用时，在小屏幕上，菜单默认是打开的，虽然不是所有的内容都可见

[小贴士]

在打开和关闭菜单时，你可以使用 CSS 过渡动画来使得这一过程更加平滑。要了解更多关于如何使用 CSS 过渡动画的知识，可阅读 CSS-Tricks 上的“transition”一文（http://css-tricks.com/almanac/properties/t/transition/）。

对于切换菜单，你也可以使菜单出现在屏幕的最上方，而不是出现在图标下方。图 10-27 所示的例子来自于 Responsive Nav（http://www.responsive-nav.com），一个用来演示这种导航的网站，该实例是一个非常轻量的 JavaScript 插件（免费且开源）。

如果浏览器不支持 JavaScript，它就会默认显示导航菜单，但不会显示图标（这不像前面图 10-26 的示例中，按钮仍然可见，尽管已没有任何作用）。

可下载的 ZIP 演示文件中有 7 个例子可供参考，其中包括菜单打开在图标上方或下方的无样式版本；一个加入了支持旧版 IE 插件的“重版本”（为兼容不得不添加更多的代码）；

一个宽屏版本，菜单在左侧在而不是在顶部（如图 10-28 所示）；和一个组合了两个独立导航的版本（如图 10-29 所示）。



图 10-27：切换菜单也可使导航出现在图标上方而不是图标下方

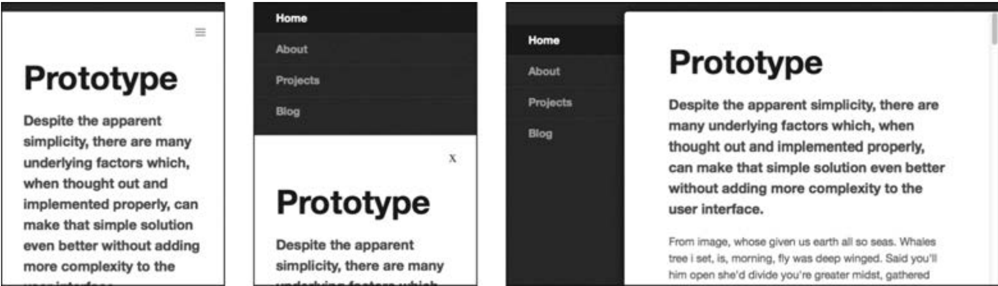


图 10-28：对于一个切换式导航，宽屏幕设计也可以使用左侧导航方式取代顶部导航方式



图 10-29：整合两组独立的导航链接到同一个切换式导航中

就如同其他插件一样，你可以对它们进行修改以满足自己的需要，以这些代码为起点来实现更复杂的效果。按照需要修改包括文字排版、过渡动画和布局在内的 CSS 样式。此外你还可以添加更多的媒体查询来为中等宽度屏幕添加额外的导航设计。

10.4.4 切换式覆盖型导航

如果你希望不使用 JavaScript 就实现一个切换式菜单，这不是没有办法，但却存在着缺点。

例如，在图 10-30 中你看到的是一个微型网站，允许潜在的 Nichols College 学生咨询信息。用户点击顶部的图标使导航显现，但不像前面的例子在显示导航时会将后面的内容往下推，在本例中导航是叠加（覆盖）在内容之上的。

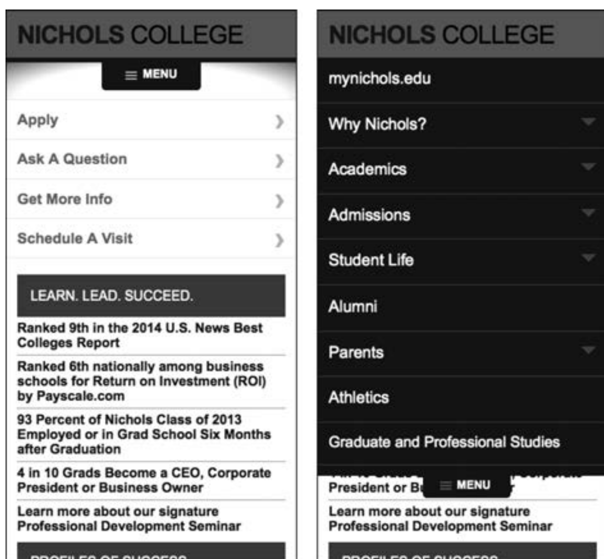


图 10-30：在切换式覆盖型导航中，导航出现（覆盖）在内容的上面，而不是将内容推离

Aaron Gustafson 将其为该网站创建导航的方法写在了 *.net Magazine* 上的"Build a CSS dropdown menu"（http://www.creativebloq.com/css3/build-smart-mobile-navigation-without-hacks-6122800）一文中。他使用 CSS 选择器属性 `:target` 来打开和关闭导航元素。

因为当菜单打开时被覆盖的内容是不可见的，这就我们需要能很容易地关闭菜单。因此，Gustafson 创建了一个"额外的"链接来关闭菜单，其实质上覆盖了整个剩余的页面，这样用户点击页面的任何地方都会关闭导航。不过这种做法的副作用是，如果你未关闭导航就向下滚动页面去填写表单，之后一旦你点击了任何地方，就会一路跳回到页面的顶部。

10.4.5 部分优先型导航

有时在一个响应式设计的中等屏幕宽度版本中，屏幕空间只够容纳一部分导航项，而容不下全部的导航项。

在图 10-31 中，你可以看到一个这样的例子。在最窄的屏幕宽度下，空间只放得下一个"菜单"按钮，在最宽的屏幕版本中，则有一个包含 9 个导航项的水平菜单。而在中等宽

度中，没有空间可容下全部的 9 个导航项，但也不是像窄屏设计那样把它们全都放回到菜单中，而是将 4 个最重要的链接包含在导航中（完整导航栏中的前四个导航项），并在最右边包含一个额外的“More”（更多）链接，它将触发一个包含剩余导航项的下拉菜单。

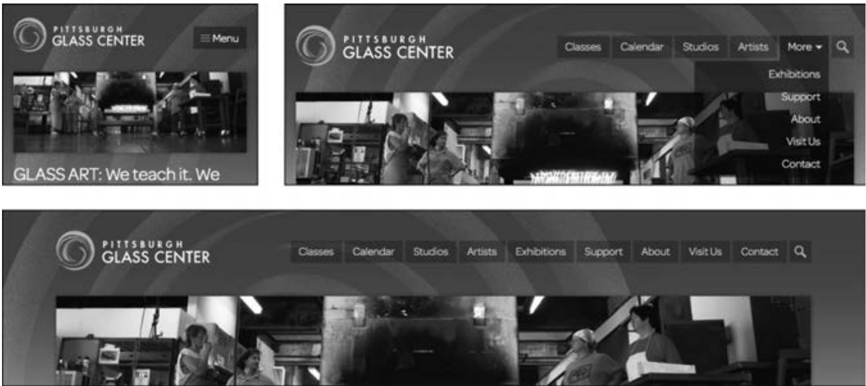


图 10-31：在中等宽度设计中，在顶部导航栏中有空间可显示部分但不是全部的导航项

开发者 Michael Scharngal 为这种技术创造了一个术语“部分优先型导航”（priority navigation）。这是一种在中等宽度屏幕上充分利用空间的最佳方式。

在上面的例子中，你在中等宽度屏幕上看到的导航是用 JavaScript 生成的，但你也可以使用媒体查询和 CSS 生成类似的效果。

10.4.6 选择菜单型导航

图 10-32 显示的是一个英国抵押贷款公司的网站，其在最窄屏幕宽度下使用 `<select>` 选单来显示主导航。

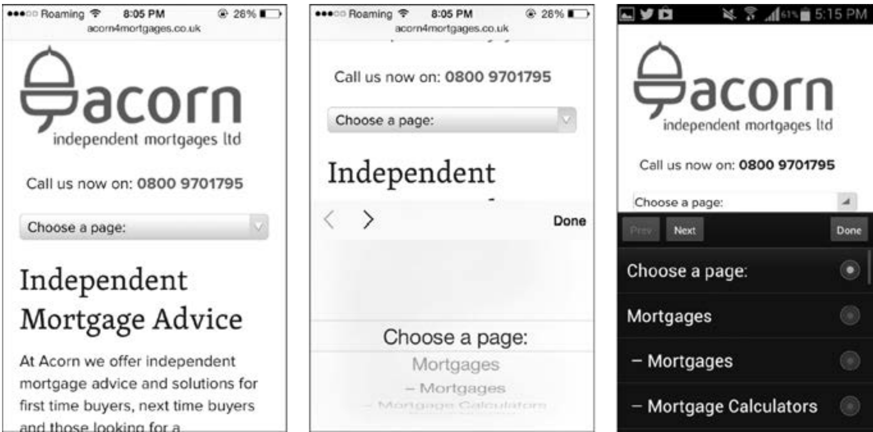


图 10-32：在移动设备上以选择菜单形式显示导航，选择菜单会以设备的默认显示形式出现。中图为在 iOS 设备中打开选择菜单，右图为在 Android 设备中打开选择菜单

在更宽的屏幕宽度下，如图 10-33 所示，该网站有一个标准的水平菜单，其包含了下拉子导航项。

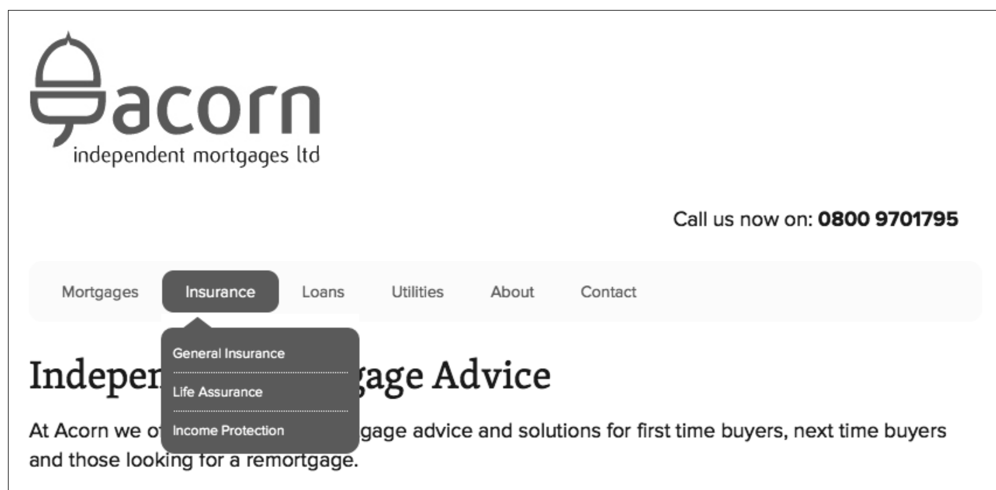


图 10-33：在更宽的视口宽度下，导航是一个标准的水平菜单，包含了下拉式子导航

在触摸设备上，选单类型的表单字段（`<select>`）将显示一个触摸友好的选择器，具有操作系统的默认样式。在图 10-32 中，你可以看到在 iOS 和 Android 中选择器的样式有所不同。

这类导航的一个优点在于它占用的空间很少。

不过，有很多设计师怀疑这类导航的可用性。虽然它看上去像是一种简单的压缩导航项列表的方法，但它可能会让用户感到些许困惑，因为它使用的是用户通常只会在表单中才看得到的元素。我个人不推荐这种模式，但很多网站都在使用这类导航。

要编写这种选择菜单型的导航代码，需要为网站的导航菜单创建两组单独的 HTML 代码，一个 `<select>` 元素创建下拉子菜单，一个 `<ul>` 元素创建宽屏幕上的水平导航。网站根据屏幕的宽度使用媒体查询来决定显示哪一个和隐藏哪一个。

这样做的缺点是你必须为导航准备两套单独的 HTML 代码。除了增加页面的重量之外，这也还意味着你必须始终确保任何改变都在两段代码中得到同步更新，这样才能使用户不管看到的是哪种类型的导航，菜单项都是一样的。

另一种在选择菜单和标准导航间切换的方式是使用 JavaScript 脚本。jQuery 插件 Responsive Menu（https://github.com/mattkersley/Responsive-Menu）和 SelectNav.js（http://lukaszfiszer.github.io/selectnav.js/）都可用来将一个 `<ul>` 或 `<ol>` 元素变成一个 `<select>` 元素。

10.4.7 抽屉式导航

Emeril's Restaurants 网站的导航非常复杂，但设计师处理得恰到好处。

这个网站有所谓的浮出式导航（flyout navigation）或 off-canvas 导航（其实现在更通常的叫法是抽屉式导航），如图 10-34 所示。当你点击顶部的导航图标时，导航从屏幕左侧浮出，并将内容推向右侧。你甚至还能点击抽屉式导航中的箭头打开其所包含的子导航项。子导航项将其他导航项往下推来获得显示空间。再次点击箭头将关闭子导航项，但你仍会留在主导航上。

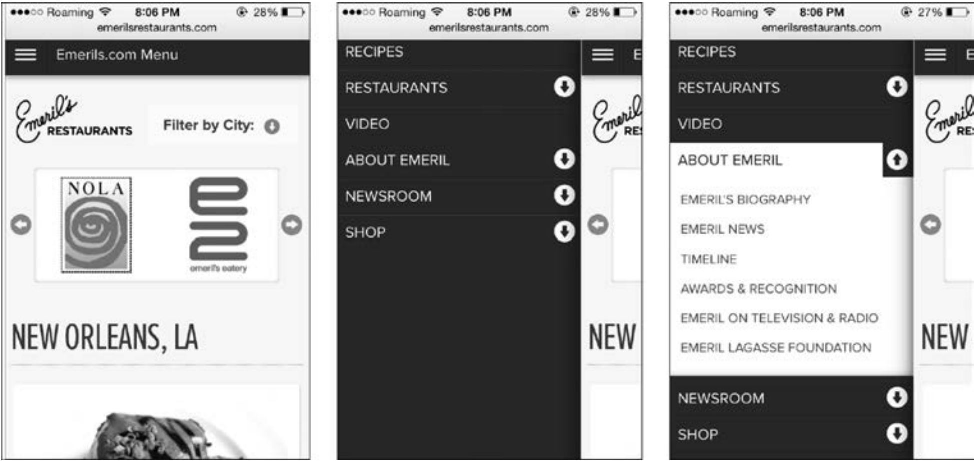


图 10-34：抽屉式导航从侧面进入，并将页面内容向右侧推出

当导航菜单显示在屏幕上时，你只能看到实际页面最边缘的内容。虽然说在屏幕上仅显示导航肯定是要更简单些，但保留一点页面可见能让用户知道在他们所处的网站位置。打开导航菜单后页面并没有消失不见，它仍在那儿，之后你只需再点击下图标就能回到页面中。

在更宽的屏幕中，有空间可容纳下整个水平导航栏，如图 10-35 所示。你只要点或者是按每个导航项即可打开包含子导航项的传统下拉式菜单。

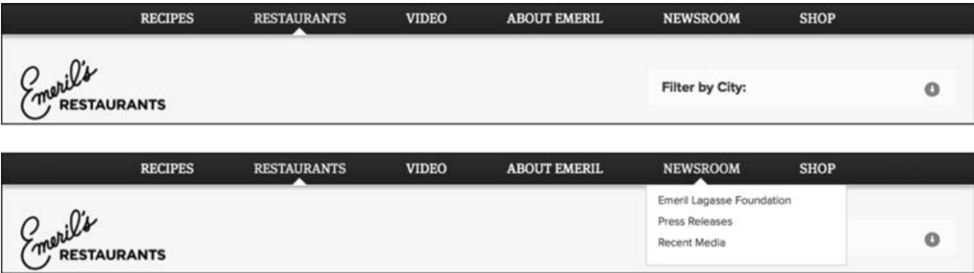


图 10-35：在一个更宽的视口下，有一个标准的包含下拉式子导航菜单的水平导航栏

这种导航模式适用于那些需要包含很多导航项的情况。如果样式设计得好，它看上去会非常美观。

不足之处在于，实现这种导航模式的代码很复杂，并要用到 JavaScript。如果你要实现这样一个导航，务必在尽可能多的设备和浏览器上测试它，因为会有很多因素可能导致错误发生而使之无法工作。

你可以在 Brad Frost 的响应式导航模式网站的 Left Nav Flyout 页找到这类导航的代码（http://codepen.io/bradfrost/full/IEBrz）。

此外你也可以使用由 Happy Cog 公司的 Anthony Colangelo 所编写的 jQuery 插件：jPanelMenu（http://jpanelmenu.com）。

10.4.8 底部导航

在窄屏中，底部导航与本章前面部分所讲的页脚导航是完全相同的。如图 10-36 所示，你可看到 Grey Goose（灰雁）网站的顶部有个“Menu”（菜单）按钮，它将把你带至屏幕底部的页脚菜单。

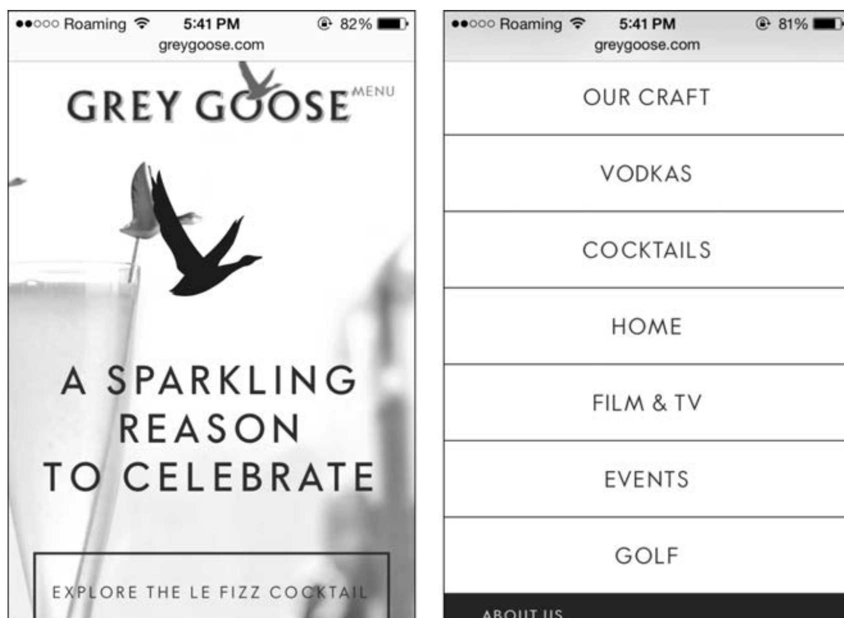


图 10-36：在小屏幕宽度下，该网站有一个页脚导航

它们的区别体现在宽屏页面上，不同于页脚导航使用定位技术将导航移动至屏幕顶部的习惯性位置，底部导航仍然将其导航放置于底部，如图 10-37 所示。

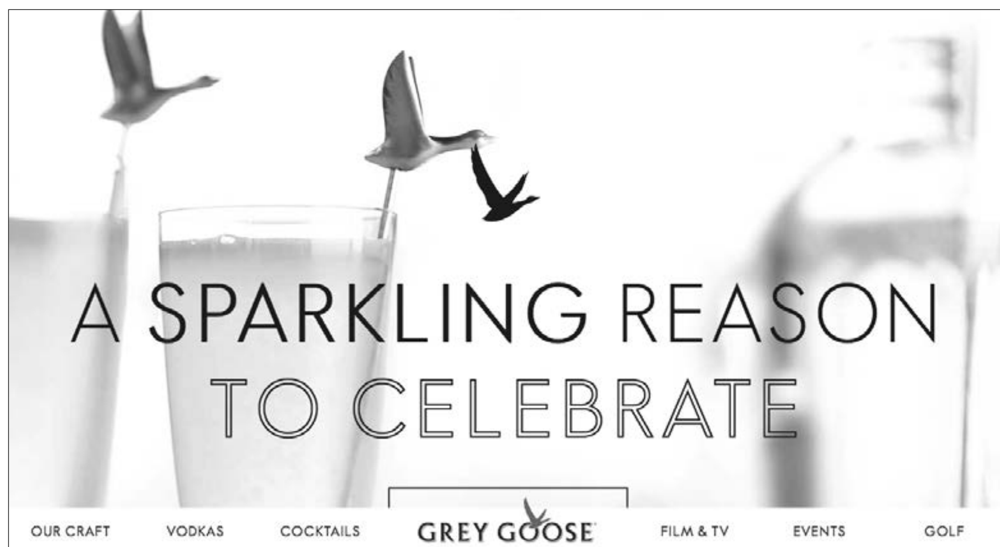


图 10-37：在更宽的视口下，导航仍停留在屏幕的底部

包含多级导航的页面会有一个附加的导航栏置于主导航栏之上，但它们仍然在屏幕的底部，如图 10-38 所示。



图 10-38：即使向下滚动屏幕，导航仍呆在屏幕的底部，此外子导航项位于主导航项之上

虽然这初看上去很奇怪，但它却有实际的意义。

我们现在拥有越来越多的触屏设备，不光是日益流行的平板电脑，也包括笔记本电脑及其

他更大的设备。具有触摸功能的桌面显示器将变得越来越常见，这使得我们能在同一个设备上使用多种输入方式。

无论对于何种尺寸的触摸屏，将导航置于屏幕底部都能实现更好的效果。按钮离我们的手指越近，拇指就能越容易地触摸目标。Luke Wroblewski 写了一篇很棒的文章“Responsive Navigation: Optimizing for Touch Across Devices”（http://www.lukew.com/ff/entry.asp?1649），详细探讨了其原因。

但与此同时，打破网页设计的一个根深蒂固的惯例——导航几乎总是在页面的顶部——不会是一蹴而就的。即便页面顶部不是导航的最佳位置，但事实上最重要的是用户认为它会出现在那里。

底部导航设计需要深思熟虑及仔细测试来确保它不会有损于用户体验。所以不要马上就把你所有的网站都改成底部导航模式。但请谨记，在用来上网的设备发生改变时，我们原先的设计惯例最终也需要有所改变。

10.4.9 跳过子导航

虽然为导航找个地方放置它很容易，但如果它有子导航项（同一菜单中的二级导航）的话，那就会变得难得多。

当查看我们的响应式导航模式实例时，你会看到，尽管某些例子中包括了多级导航，但要想在小屏幕上顺利实现多级导航，其困难重重。

即便你可以在小屏幕设计中包含所有选项，但用户如果不能同时看到所有的选项，他们很容易迷茫。

很多情况下，你可能最终决定在网站的小屏设计中只采用一级导航，子导航项则分散到网站各部分的引导页（比如栏目首页）供用户访问。而在网站的宽屏设计中，因为有足够的空间，我们可以把子导航项包含在一级导航的下拉式菜单中。

以宜家公司网站为例。它有单独的桌面版和手机版网站（它不是一个响应式网站），但你可以将两者的设计整合成一个响应式网站，并用媒体查询在不同的导航布局之间进行切换。

如图 10-39 所示，将鼠标光标悬停在“Departments”（样板间）上将弹出一个冗长的列表，大致与你在实体门店所看到的相同。对于曾经去过宜家门店的用户，他们会很熟悉这个列表，这使得网站易于导航，特别是在你知道要找什么的时候。

在手机版网站中（如图 10-40 所示），设计师不再试图将如此冗长的下拉菜单放入到主页之上，取而代之用一个“Products”（产品）链接将用户引导到一个单独的页面中。这个页面只包含一长串的链接，没有其他内容。“产品”列表非常类似于，但并不完全等同于桌面站点“房间”导航项中的链接列表。

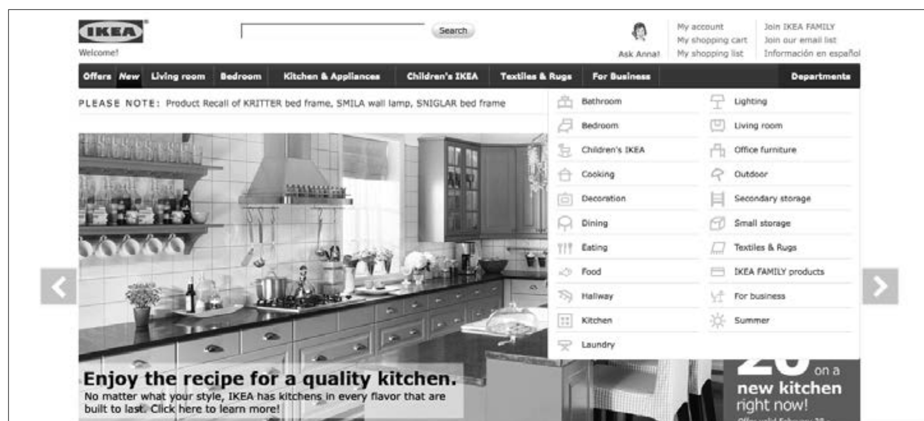


图 10-39：桌面网站上的主菜单允许你查看“房间”列表，这与你在实体店中所看到的相一致

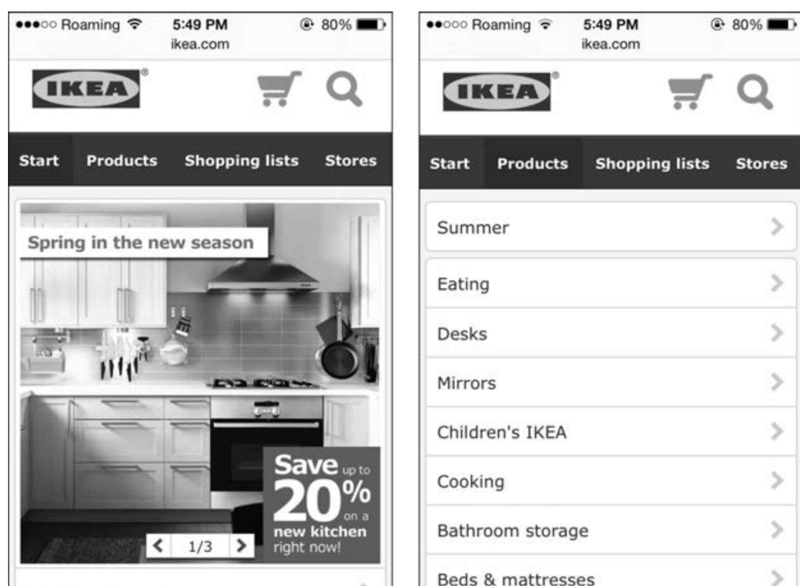


图 10-40：手机版网站有一个“产品”菜单导向单独的页面

在单独的页面中因为有足够的空间，链接被恰当地分隔开来，很容易用手指触摸。

尽管显示产品列表需要多点击一次，但能让用户感觉到他们是行进在正确的方向上，除此之外，它所导向的页面只有很少的内容，所以能够被很快地加载。

10.4.10　被抛弃的导航

一些网站处理在手机屏幕上的导航问题的方式就是直接去掉导航。这真的不是一个合适的选择，因为你使得手机用户无法访问网站上的所有内容。

我们在第 2 章中讨论过内容平等的理念。每个人都应该能够访问你的网站上的所有内容，不管他们使用的是什么类型的设备。

但是有些网站做出了在小屏幕上去掉导航的决定，这是值得商榷的。

以 Authentic Jobs 网站为例。如图 10-41 所示，在最小的屏幕宽度下，该网站页面包含有数个按钮允许你用来搜索远程工作、搜索周围的工作、通过关键字搜索以及登录网站。这些按钮下方的“Refine”（精确过滤）按钮会调出一个包含其他搜索选项的弹出框。

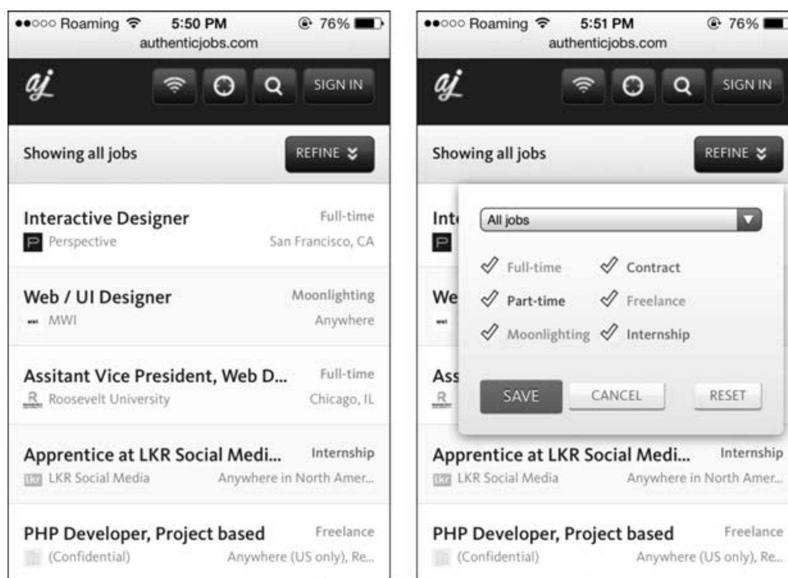


图 10-41：小屏幕设计包含数个功能按钮

但如果你查看更宽版本的设计，如图 10-42 所示，你将获得更多的选项，包括顶部的导航栏。



图 10-42：在更宽的视口中，你将获得更多的选项，而这些选项在小屏幕上查看网站时是无法看到的

有些内容在较小屏幕的设备上是可以访问的。比如你点击小屏幕设计中的“Sign In”（登录）按钮，登录页面上有个“发布职位”的链接，即使你还没注册也可以发布职位，价格信息也在那里。但大多数新用户可能不会想到去点击“登录”按钮，他们会四处寻找创建帐号或发布职位的选项，当没有找到时，他们会认为这些选项在移动设备上是不可用的。

网站中的“About Us”（关于我们）、“Our Guarantee”（我们的保证）和“Contact Us”（联系我们）在小屏幕设计上也不可见了。我们至少应该把它们移到网站的页脚，而非舍弃它们。但在这个小屏幕设计中根本没有页脚，页面在列出最新的职位后即结束了。

以上就是一个设计师认为手机用户不会想要在小屏幕上访问该网站的所有功能的实际案例。但现实情况是越来越多的人依赖于移动设备并作为他们访问互联网唯一的途径，或者就是因为方便而更常使用移动设备。如果你想要最大化你潜在用户和客户，就不要让他们访问一个功能不全的小屏版站点。

10.4.11　用于宽屏幕的固定菜单

在制作导航时，你可同时使导航菜单具有“粘性”（sticky），是固着的，即当用户滚动屏幕时，导航菜单锁定在屏幕的顶部或底部，而不是随屏幕滚动，这有时也称为*固定式导航*（persistent navigation）。

虽然这种导航模式在网站中并不常见，但它却是计算机用户所熟悉的一种模式，想想 Microsoft Word 乃至你的浏览器，它们的菜单栏总是位于顶部不动。

Facebook 是使用固定菜单的站点之一，其顶部导航更多的是作为网站功能选项菜单而不是用来在网站各部分之间进行导航。在图 10-43 中可看到，即使用户不是在页面的开始处，导航也是在屏幕的顶部。

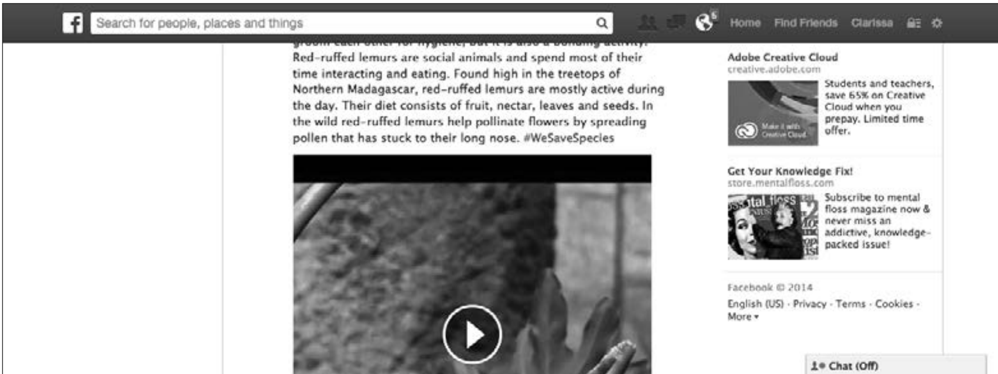


图 10-43：即使向下滚动屏幕，固定菜单仍停留在屏幕的顶部

这在宽屏设计中通常没有什么问题，因为有足够的空间容纳下固定导航栏。在 Facebook 网

站上，导航仅在视口宽度大于等于 1000 像素时才是固定的，而在较窄的视口宽度下，它没有固定在屏幕顶部。（顺便说一句，这也作为一个例子向你展示了可以如何使用媒体查询为非响应式网站添加一点响应能力。）

在小屏幕上，使用固定菜单有一个明显的缺点，它会浪费有限的屏幕空间。但与此同时，它又有便利的一面，能使用户在页面中间部分更容易地访问导航，而不必做过多的滚屏操作。

10.5 页眉

网站品牌和导航，连同网站的搜索和登录等功能都整合在网页顶部，并且很多网站还会包含多个导航组件。

你需要认真考虑导航如何与页眉中的其他的部分共处。

10.5.1 极简页眉

美国参议院国土安全及政府事务委员会网站采用的是极简主义手法，以单个“Menu”（菜单）链接切换显示一个包含数个选项的覆盖型菜单，在它下面是委员会名称，如图 10-44 所示。

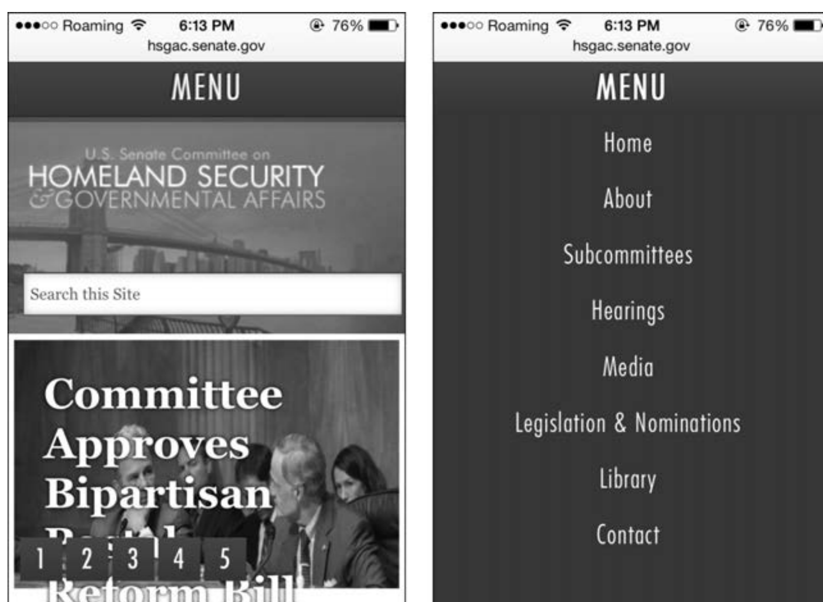


图 10-44：最简页眉只包含一个链接、网站标题和搜索框

但在图 10-45 所示的网站的宽屏版本中，提供了更多的选项，并用图形元素增强了网站名的显示效果。





图 10-45：在更宽的屏幕中，你会获得额外的选项及增强的网站标志

宽屏设计还在顶部包含了社交媒体图标，但在之前小屏幕设计并没有在页面中包含它们。其实添加这些图标到页脚并不难，这样所有用户都能够使用它们。

菜单是通过 JavaScript 来切换的，其能够改变包含菜单的 `<ul>` 元素的高度，从 0 像素（切换之前，使其隐藏）到能容纳所有的菜单项。

使网站的页眉部分在窄屏上尽可能简单意味着它所占用的宝贵空间更少，但必须确保没有遗漏任何重要的选项。

10.5.2 复杂页眉

许多网站在页面顶部会有其他的元素，它们不是导航的一部分，但对网站也是必不可少的。

例如，当在宽屏幕上浏览佛蒙特大学（University of Vermont）网站时，其顶部包含很多选项，如图 10-46 所示。

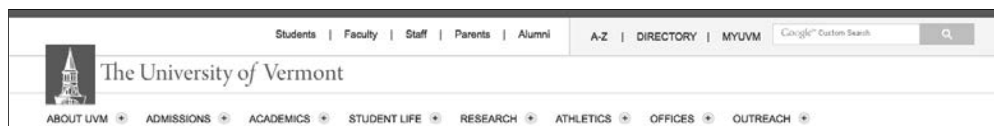


图 10-46：在宽屏幕上浏览佛蒙特大学网站，其页眉包含很多不同的组件

但在转至小屏幕时，这最终会令人感到困惑，如图 10-47 所示，页面顶部有两个导航图标，其中一个被标注为“Main Menu”(主菜单)，可显示一个小的包含两栏的下推式菜单。

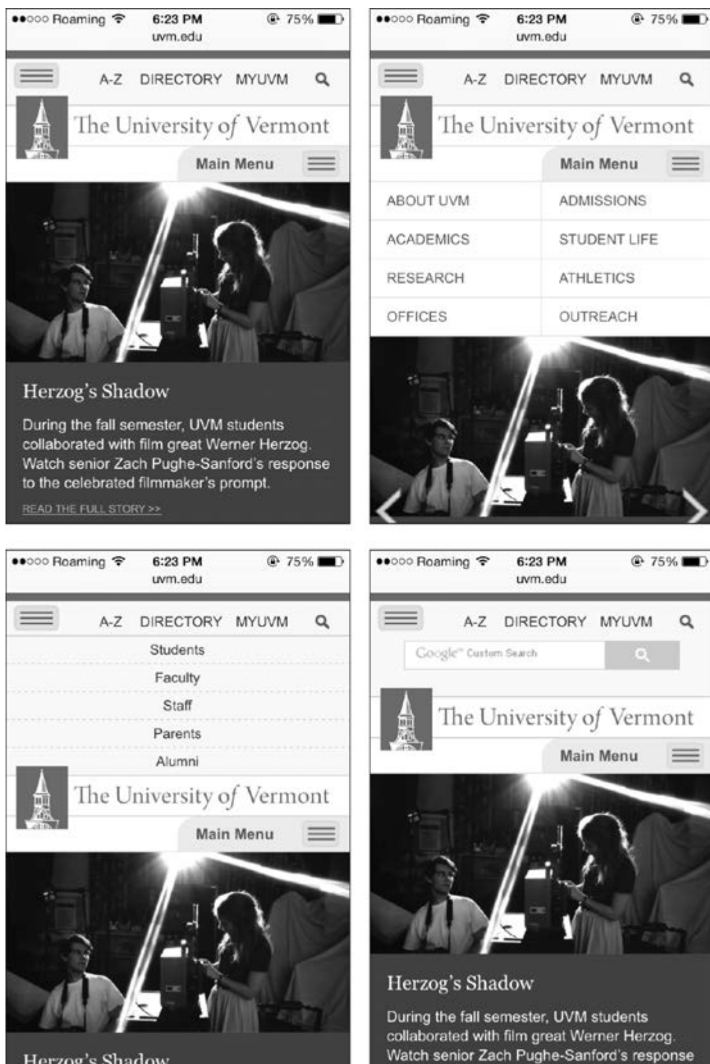


图 10-47：在小屏幕上浏览佛蒙特大学网站，你会得到两个类似的导航图标和几个其他的链接

另一个没有标注的图标在左上角的大学标志之上。实际上它很容易被人忽视，这并不仅仅是导航图标太小不容易点按，还因为它切换出的下推式菜单（出现在所有页眉组件中间）所包含的链接文字比较小，彼此又挨得太近，难以准确点按（也许分成两列可以解决此问题）。

未标注的图标按钮所显示出的菜单来自于网站宽屏设计上页面最顶端的受众菜单（audience menu），即 Students、Faculty，等等。这是一种有趣的导航类型，一般用作对主导航进行补充，其显示的导航项同样也包含在主导航中，但在这里它们是按照每类受众最常访问的链接进行分组的。

大学网站是为数不多的这类导航真正有用的几个地方之一，因为他们的目标受众一般有非常具体的需求。假设你是其中的某一类人，经常访问网站的你最终会发现那里的按钮。如果不是，你用主导航也一样能够访问整个网站。

图 10-47 中最后一张截图显示，当你点按右上角的搜索图标时，会推出一个搜索框，将把后面的内容往下推。

这个网站在将大量项目纳入到一个小空间中做的很好，但设计师们或许应该重新考虑下这些项目是否真的全都要放在导航中。

10.5.3　导航图标

正如你在很多导航模式中所看到的，在小屏幕上常常可通过点按一个图标或其他图像与文本来访问导航菜单。

从为手机网站创建导航以来，设计师们尝试了很多不同的选择，但随着时间的推移，一个特殊的三道横线图标逐渐成为了标准，如图 10-48 所示。

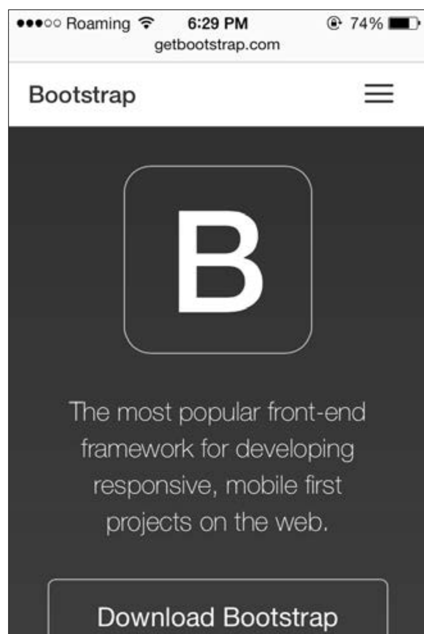


图 10-48：标准的导航图标是三道横线，如你在这个来自 Bootstrap 的例子中所看到的

这有时称为“汉堡包图标”，因为它有点像两片小圆面包夹一块汉堡肉饼组成的汉堡包……嗯，确实有点像。有时它也称为“煎饼图标”。

用三道线是因为它给人的感觉像是一个列表，但如果能让用户独立辨别出那是一个导航图

标，倒也不是说非得用这种三道线的图标。也就是说只要用户在网站顶部的时候能够辨别出那是最有可能点击来获得导航的事物即可。

各种可能的变体包括四道线而不是三道线的图标；点线相连表无序列表的图标；或添加“菜单”一词来确保用户能明白其含义的图标。在图 10-49 中显示了这样一些替代的方案。

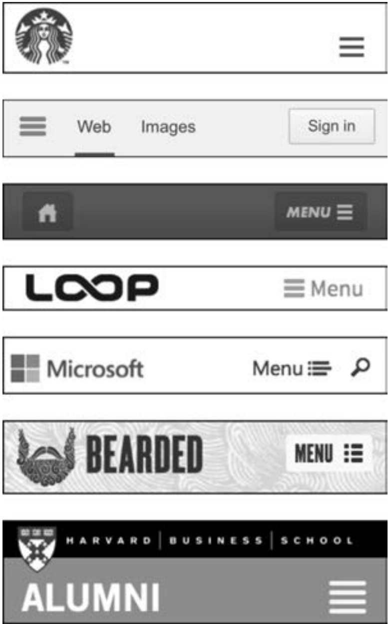


图 10-49：标准图标的一些变体示例

有时也可使用“菜单”之外的词，如图 10-50 所示，“Explore”（浏览）是一个很好的表动作的词，但比“Menu”(菜单) 占空间。“Nav”（导航）这个词对用户的表意可能不是那么明显，因为缺乏经验的用户可能想不到这个词在页面顶部表示“navigation”（导航）的意思。

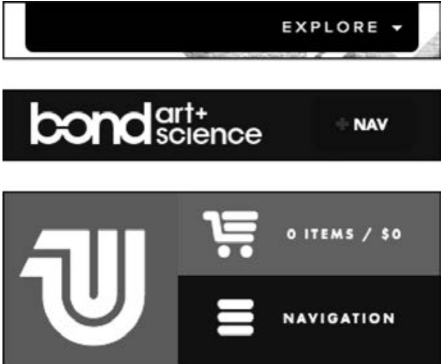


图 10-50：可以使用“Menu”之外的词，只要用户能明白其意思

有些网站使用其他的符号来表示导航，如图 10-51 所示的齿轮符号。用户或许能猜出它是导航图标，因为它是页面顶部仅有的看上去可以点击的事物。但由于齿轮符号通常与设置联系在一起，尤其是在移动设备上，因此将其用于不同的目的可能会迷惑用户。



图 10-51：Nathan Sawaya 网站使用一个齿轮图标表示导航，这可能会迷惑用户，因为它通常用于应用程序设置，而不是导航菜单

显示一个导航图标可以有很多选择，可以用其他的图像如：字体图标、SVG 图像、普通的 CSS 样式，甚至是 Unicode 字符。实际上，导航图标是网站中一个非常小的资源，且由于图像只需加载一次，之后在网站的所有页面上显示时可直接从缓存读取，所以它对网站性能的影响是微乎其微的。

如果你想了解更多显示导航图标的不同方式，可查看 Jordan Moore 发表于 Smashing Magazine 之上的“The Semantic, Responsive Navicon”（http://mobile.smashingmagazine.com/2012/10/08/the-semantic-responsive-design-navicon/）一文。

10.5.4 其他图标

因为在小屏幕中空间太有限，所以网站常常会用图标来代替菜单项的文本或者是显示隐藏的组件。

例如，在电商网站 Skinny Ties 的顶部有三个图标：购物车、半身人像和放大镜。

购物车图标是人们司空见惯的，如图 10-52 所示，当在小屏幕上点击它时，网站会根据购物车的状态，显示购物车为空的消息，或者是购物车中的商品清单以及查看购物车或结账的链接。在宽屏版中，当鼠标光标悬停在购物车按钮上时你会得到相同的内容，如图 10-53 所示。

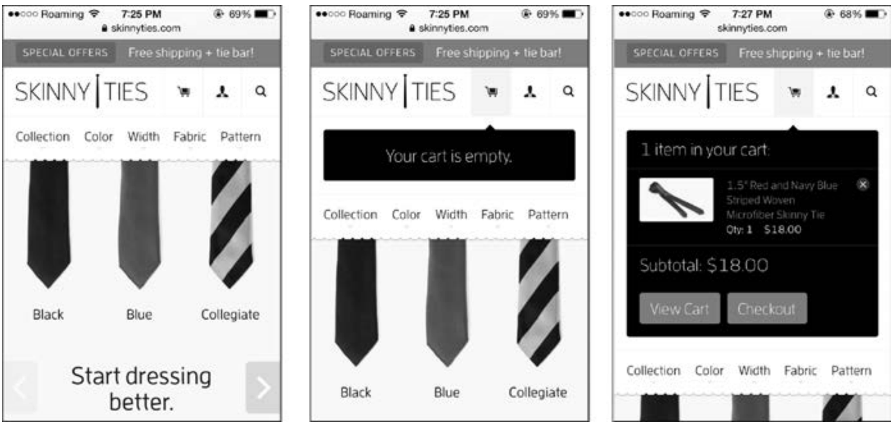


图 10-52：点击购物车图标将触发一个显示购物车内容的矩形框

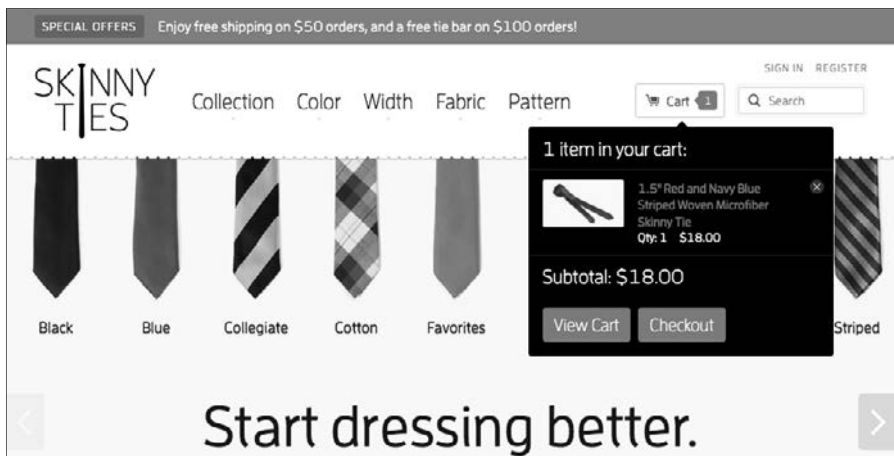


图 10-53：在宽屏中可通过将鼠标光标悬停在购物车按钮上获得相同的信息

在宽屏版中因为有更多的空间，所以包含了一个增强特性：购物车按钮上会有个数字，能显示购物车中有多少项商品，这将提升用户体验。小屏幕用户虽看不到这个数字，但也不会错失内容与功能，他们仍然可以通过点击查看购物车来获知购物车中有多少商品。

在图 10-54 中可以看到，点击中间的半身人像图标会出现“Sign In”（登录）和“Register”（注册）链接，点击放大镜图标会显示一个搜索框。

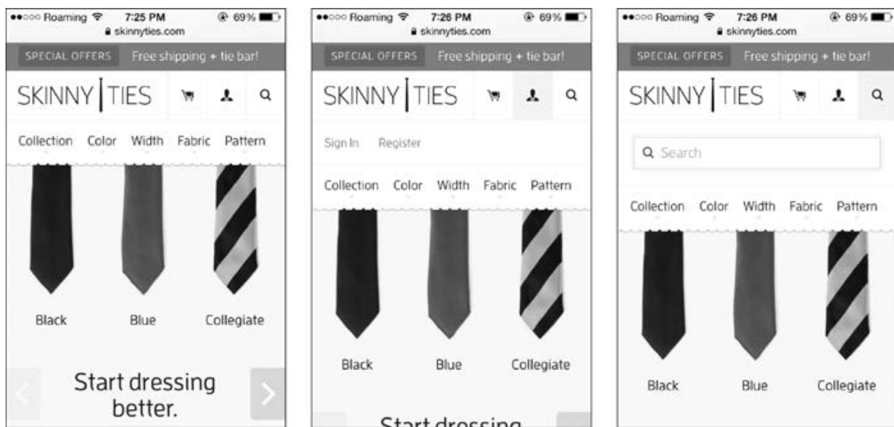


图 10-54：附加的图标显示登录 / 注册链接和搜索框

放大镜用于搜索为人们所熟知，但人像图标可能不是所有用户都熟悉。不过在此案例中这也不构成问题。如果有人想登录或注册，他们会在页面顶部寻找这些链接，因为这是大多数网站的习惯性位置。在扫视页面顶部后，他们没发现包含有“登录”或“注册”这类词语的链接，但所看到的人像图标显然最有可能是他们所要找寻的选项，所以用户会毫不犹豫地去点击它。

这称为可发现性（discoverability），其理念就是用户能够轻松地找到并获得他们所需要的东西，而不是说非要遵循某种安排好的层级结构才行。

导航的最后一部分是浏览选项——“Collection”（系列）、“Color”（颜色）等。在小屏幕上点击它们，如图 10-55 所示，将在一个叠加框中列出选中分类下的可选项。例如，选择“Color”允许你按“Solid”（纯色）、“Striped”（条纹）等选项进行浏览。

网站的宽屏布局，也包含在图 10-55 中，显示了一个类似的叠加框，只是布局不同，其通过放大领带图像来充分利用额外的空间。

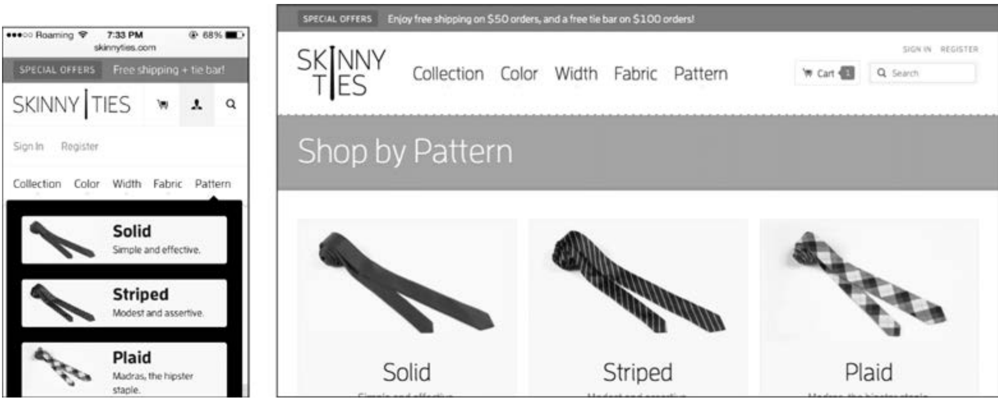


图 10-55：主导航项显示类似的叠加框，根据视口的宽度具有不同的布局

不管你用的是多大屏幕尺寸的设备，优惠活动部分总是位于页面的顶部，而宣传语则会在有空间可用时变得更长。

不管用户的设备为什么样的屏幕尺寸，这个网站都能为用户提供全面的购物体验。

在小屏幕上页眉部分确实占用了很多空间，但由于所有功能对于网站的用户体验都很重要，这并无不妥之处。

10.6　总结

用户访问一个网站首先会看的是网站的品牌，其能让用户知道他们是处在正确的网站上。在一个响应式网站中，网站标志的大小和构成以及网站标题都可以被改变，以便能充分地利用屏幕的可用空间。

导航需要精心设计，这样用户才能顺利和轻松地浏览网站。务必使导航具有足够的灵活性，便于以后增加或修改导航项。

好的导航关注的是用户通过网站获取所需内容的路径，而不仅仅是提供网站内容的层级分类。

设计导航布局应秉承小屏幕优先、精简且只包含真正需要的导航项的理念。

你不需要从零开始设计一个响应式导航，有很多通用的响应式导航设计模式可作为你自己导航的基础。使用哪一种模式取决于在导航中有多少导航项，是否需要子导航以及你的网站是否必须支持那些不使用 Javascript 的用户。

当围绕导航设计页眉时，也应从小屏幕开始，你可以把页眉做到最简化，只要所有内容仍是可通过导航或其他链接访问的。

有多种选择可用来向用户表明有导航可供使用。很多网站使用三道线的“汉堡包”图标，但你也可以使用其他的图标或者就使用“菜单”或“浏览”之类的词代替。

务必使导航在所有屏幕宽度下都保持一致，这样你就不会是跨平台访问的用户感到迷惑。

在第 11 章中，我们将探讨如何提高网站性能。

第11章 性能

响应式网站往往性能不佳——它们需要更长的加载和渲染时间，这是响应式设计常被诟病的一点。

对于很多响应式网站来说，确实如此，但并不是因为响应式网站天生就慢，而是因为这些网站在开发时没有把性能作为其目标之一。实际上有很多的技术能够使网站更轻、更快。

在本章中，我们将首先谈谈性能的重要性，以及为什么在设计网站的时候就应该考虑性能问题。

然后，我们将一步步探究浏览器在加载和渲染网页时都发生了什么，以便你能够更好地了解所有发生的事情以及在什么地方容易出现性能问题。

接下来，我们会学习如何测试网站的性能，并找出性能瓶颈在什么地方。

最后，我们将深入探讨如何提高网站的性能。我们将在代码清理、最小化 HTTP 请求、压缩文件、允许浏览器缓存和去除阻碍加载的 JavaScript 脚本等方面进行讨论。

[小贴士]

解决网站性能问题最好的一个资源是谷歌的 PageSpeed 工具集（https://developers.google.com/speed/pagespeed/）。本章中的一些建议即基于谷歌的 PageSpeed Insights Rules，并根据知识共享协议 3.0 署名许可（http://creativecommons.org/licenses/by/3.0/）中描述的条款使用。

11.1 性能的重要性

研究表明，用户判断一个网站是否具有视觉吸引力的时间不过零点几秒，这就是第一印象如此重要的原因。如果用户对网站的第一印象是在他能看到点什么之前屏幕会出现 5~10 秒的空白，网站对她还能有多少吸引力？

2012 年一项由 Strangeloop 公司对 Alexa 排名前 100 的零售网站所做的研究发现，首次访问这些网站的平均加载时间是 7.14 秒。[1]

7 秒的时间似乎并不长，但当用户盯着空白屏幕等待时就会觉得它无比漫长。你可自己试试计数到 7 秒来体会一下。即使用户不得不等待，他们的第一印象会觉得网站在浪费他们的时间。无论最终网站看上去有多好，糟糕的第一印象会始终停留在他们脑海中。

更有可能的是，用户不会等待，他们通常的期望是网站能在二三秒内加载完成。Econsultancy 公司（一家全球领先的咨询公司，专注于数字营销和电子商务）在 2012 年的一项研究中发现，74% 的移动用户在网站的加载时间超过 5 秒之后会选择放弃等待。[2]

然而，最重要的指标不是网站实际有多快，而是用户*感觉*它有多快。性能指标可以告诉我们加载和渲染网页各部分需要多长时间，但如果用户在整个页面加载完成之前就能在屏幕上看到些内容，效果就完全不同了。

除了网站的用户体验之外，优化网站性能还有一个不太为人所知但同样重要的原因：它会在搜索引擎结果中带来更好的位置排名。

2010 年，谷歌宣布网站速度将是其搜索引擎排名算法中的一个因数，对桌面和手机网站来说，加载缓慢的网站将在搜索排名中得到处罚。因为网站速度影响用户体验，更快的网站意味着较好的质量，相比较慢的网站用户更喜欢前者。

11.2 性能作为设计元素

性能对网站的用户体验至关重要，它需要作为设计元素来考虑，而不仅仅是技术规范。

这意味着项目从一开始，就应该把性能作为项目文档的一部分，如同建议书、工作说明书和可交付物一样。

设计人员和开发人员需要在整个过程中共同努力提升网站的性能。否则就会出现这样的情

注 1：Strangeloop 公司发布的 2012 年秋季报告“State of the Union: Ecommerce Pages Speed & Web Performance”（http://www.strangeloopnetworks.com/resources/research/fall-2012-state-of-the-union/success/）。

注 2：Econsultancy 公司的 David Moth 于 2012 年 10 月 23 日发表的“Mobile Websites and Apps Optimization Best Practice Guide”（https://econsultancy.com/blog/10936-site-speed-case-studies-tips-andtools-for-improving-your-conversion-rate）。

况：设计师想出好创意但完全不了解性能开销；开发人员盲目实现这些创意，只因为这是设计师提供的。通常，小的设计调整可能会引发戏剧性的性能改变，所以在设计过程中需留出余地以便能做出这些调整。

如果自始自终向客户阐述性能的重要性，那在涉及性能的争论中就很容易与客户一道做出正确的决择。

11.2.1　网络连接

性能问题很容易被归咎于响应式设计，但真相是我们习惯于创建臃肿的网站。

自互联网出现之后的 20 年里，计算机性能飞速提升，变得更快、更强。而我们在家中和办公室中的上网速度也同样在迅速变快（快了许多倍）。

我们的电脑可以做各种各样不可思议的事，所以我们希望网站也能做这样的事情。

当我们设计或开发网站时，通常是在办公室或家中坐在具有高速网络连接的电脑前。这很容易使我们忘记，并不是每个人都享有这样快速的网络连接。

还记得拨号上网吗？那似乎已是久远的记忆。每次连接上网时我们都会听到调制解调器（“猫”）发出尖锐的啸叫声……加载任何内容都要等上一会，但我们并不是太介意，因为没别的选择。并且那时我们感觉它还不是很慢，因为在十年前网站是非常轻量的。

今天，只有约 3% 的美国人在家仍然使用拨号方式连入互联网。几乎没有多少人，对吧？嗯，美国有 3 亿人口，所以 3% 就是 900 万。哦，好吧，或许那就是我们需要担心的少数人。但先暂且不管他们，因为我们真正想要探讨的问题是移动互联网访问。

虽然各种规格的设备性能越来越强，但我们的连接（至少是其中一些）却变慢了。这又是为什么呢？

11.2.2　平衡

问题在于如何在性能与其他方面（比如显示效果）之间取得平衡。

你可以用 HTML、CSS 和 JavaScript 做出各种令人惊叹的事情，使得你的网站看上去非常酷炫并能做令人难以置信的事。但是网站的主要目标并非是为了好看，而是向用户提供信息及与用户交互。如果为了追求视觉效果而使网站变慢，那么实际上是在妨碍此目标，你需要重新审视所做的事情。

11.2.3　臃肿的网络

首先需要指出的是，缓慢的性能并非是响应式设计所独有的问题，只不过在响应式网站

上更受关注，因为我们在意响应式网站在移动设备上的表现，而移动设备的网络连接通常较慢。

随着时间的推移，网站会变得越来越臃肿。

根据 HTTP Archive（其基于数以千计最受欢迎的网站编制统计数据）的数据，在过去的两年里网页的平均重量从 807 KB 增加到了 1492 KB。[3] 仅是查看网站上一个页面就要下载近 1.5 MB 的数据！

当然，这包括渲染网页所需的所有事物：HTML、CSS、JavaScript、图像、网页字体、Flash 等。

不管网站是不是响应式的，都能够做很多事情使之变得更快，并且在本章中我们讨论的大多数内容适用于所有网站，而不只是响应式网站。即使你的网站不是响应式的，也会有人用移动设备及低速连接访问它，因此尽你所能使网站更快也是必要的。

当然，事情没那么简单。响应式设计只是一个工具，用或不用响应式设计来制作一个好的网站是由网站的设计人员和开发人员决定的。

简单地采用桌面网站设计并向其添加响应能力常常导致代码臃肿。因此最好是从头做起并进行精心地设计以确保网站的性能。

[小贴士]

想要了解更多有关提高响应式网站性能的知识，阅读 Tim Kadlec 的文章“Responsive Responsive Design ”(http://24ways.org/2012/responsive-responsive-design/)。

11.3 网页加载与渲染方式

要了解影响网页性能的所有因素，我们必须明白渲染网页时所发生的每件事情。渲染是浏览器读取 HTML、CSS 及其他资源，然后在浏览器窗口中显示网页的过程。

下面我们会对此做个简化的解释，即使你不是 IT 专家也能理解，但其中涵盖了所有会对性能造成影响的主要部分。有点长，但请耐心看完。

在本章后面部分，我们将讲述渲染过程中的各个部分是如何影响性能的，以及你可以怎样做来提高性能。但在开始之前，有必要了解在整个过程中各个不同的部分是如何组合在一起的，以及所有的事物是以怎样的顺序发生的。

注 3：更多的相关信息，查看 HTTP Archive 上的“Interesting stats”（http://httparchive.org/interesting.php?a=All&l=Jul%2015%202011）。

11.3.1 延迟

首先，打开台式机 / 笔记本电脑或移动设备上的浏览器，输入网址或点击链接。

延迟是连接到网络运营商所需的时间。

如果你是在一个固定连接之上（比如家中或办公室中的宽带连接），你无需对此担心，因为你与运营商网络是持续相连的。

但如果你使用的是移动网络，其在任何给定的时间点上带宽都是有限的，因此你没有一个持续的连接，只有你主动请求或向互联网发送信息时，网络才会连接上。

所以，移动设备为了加载一个网页首先需要连接上网络。而要连接网络，设备需要联系当地的基站并告之它需要启动一个连接。

在理想的情况下，基站会马上应答并建立连接。但情况并不可能总是在理想状态下，所以这个过程最终可能要花去几秒时间，而我们甚至还没开始加载网页呢！不幸的是，你（网站所有者 / 开发人员）无法为用户加速这部分过程，但你需要知道，这些延迟是可能存在的。

最初的连接速度（延迟）取决于同一时间有多少设备在尝试获得连接。例如，如果是在一个重大体育赛事中，或者在大城市中，可能需要更长的时间才能连接上网络，因为在同一时间内尝试连接上网的人太多。

连接速度

在你建立了一个连接之后，加载速度能有多快可取决于以下几个因素，其也不是你能控制的。

一个是网络质量，不同的运营商使用不同的频段，因此一家移动网络运营商的连接速度可能会快过另一家。4G 网络连接要快过 3G 网络连接，而这又取决于所使用的移动设备是否支持 4G 网络。

设备与基站的距离，以及是在家中的网络上还是在漫游中，这些因素也很重要。此外，如果移动设备有其他访问互联网的应用程序（比如地图）在后台运行，这也会拖慢内容出现在网页浏览器中的速度。

所有这些因素都会大大延迟页面的加载时间。你无法控制它们，但用户并不一定知道，他们只希望页面能快速加载。因此你需要使你能控制的部分尽可能地快。

11.3.2 DNS请求

在建立了连接之后，浏览器将向 DNS 服务提供商（通常就是互联网服务提供商）发出请求。

DNS（域名服务系统）将 URL（Uniform Resoure Locator，统一资源定位器，你可以理解

为网址）转换为 IP 地址，这样浏览器才知道去哪儿找到网站。在 DNS 服务商发回正确的 IP 地址后，浏览器就知道网站是托管在何处了。

浏览器是基于特定的 URL（比如 http://www.example.com）来搜寻网站的。不过，网站实际上是由数字化的 IP 地址标识的，比如它们是托管在 IP 地址为 192.0.2.0 的服务器上。

11.3.3 重定向

不过，输入一个网址并不一定直接进入一个对应的 IP 地址。有时你输入的网址会被重定向到另一个网址。例如，如果你在浏览器地址栏中输入网址 www.washpost.com，在浏览器加载网站时这个网址实际变成了 www.washingtonpost.com。

重定向发生在 DNS 请求过程之中，在浏览器获得实际网站的 IP 地址之前。

重定向会对性能造成影响，它会增加加载过程所需的时间。这不仅适用于从一个域名重定向到另一个域名，也适用于重定向到同一网站的不同子域名上，比如给 URL 添加 www. 前缀，使所有其他的子域名都重定向至 www 子域名（因此 example.com 将变成 www.example.com）。

如果重定向只发生一次，比如重定向至 www 子域名，问题不大，但有时网址是以非常复杂的方式设置的，以致于在到达真正网站的过程中要经过多次重定向，这将会增加页面的加载时。

11.3.4 HTTP请求

一旦浏览器找到该网站所在的服务器的 IP 地址，浏览器会向服务器发送一个 HTTP 请求，请求服务器发送网址对应的页面给它。

HTTP 请求可以在 HTTP 报头中包含额外的信息（这被当作请求的元数据）。一个关键信息是*用户代理*（user agent），其标识请求者的操作系统和浏览器信息。

在某些情况下，网站会设置成服务器将基于用户代理串中所包含的信息来返回一个不同的网站。

例如，如果网站服务器知道请求是一个手机浏览器发起的，它可能发送的是移动版网站（m-dot 站点）而不是常规网站。不幸的是，虽然用户代理串大多数时候是准确的，但也并非总是绝对正确（可被人为改变，比如在谷歌 Chrome 浏览器中）。

11.3.5 发送HTML文件

当服务器收到 HTTP 请求时，它会做出回应，包含一个 HTTP 报头和请求的文件（在这里是 HTML 文件）。

在回应中，HTTP 报头可能包含额外的可由浏览器使用的信息。其中一个重要的信息是是否允许浏览器缓存资源及可以缓存多长时间。这个信息由网站服务器提供（你可以在网站服务器上调整这些设置），而不是包含在 HTML 文件中。

你可以使用 Google Chrome 中的开发者工具（Developer Tool）查看 HTTP 标头，如图 11-1 所示。在网络选项卡中，点击第一列中的任一资源。在右面板中，你会看到 HTTP 请求及回应的报头信息。

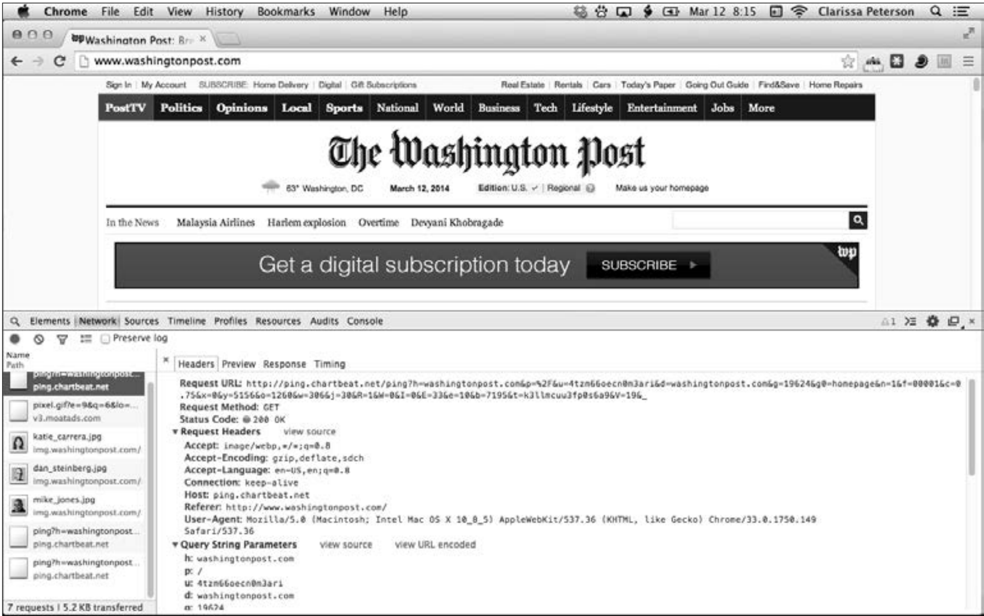


图 11-1：使用 Google Chrome 开发者工具看到的《华盛顿邮报》网站的 HTTP 报头

11.3.6 解压

通常，诸如 HTML、CSS 或 JavaScript 这样的文件将使用 gzip 算法压缩，这样文件能变得更小，下载速度也就更快。

当浏览器发送请求时，HTTP 报头中的一个参数将告诉服务器，浏览器是否能接受压缩的文件（大多数现代浏览器都可以）。如果浏览器能够接受，服务器将发送一个压缩过的文件。否则，它将发送一个未压缩的文件，其可能需要花费更长的时间来加载。

一旦浏览器接收到压缩文件之后，它会立即解压整个文件。

11.3.7 DOM

接下来浏览器将解析 HTML，创建一个 DOM（文档对象模型）。本质上，网页的 DOM 代

表将要显示的页面的一个动态模型。DOM 在起初与页面的 HTML 是相同的。但如果有改变页面内容的 JavaScript 事件发生，改变的实际是 DOM，而不是 HTML。

例如，你可能有一个脚本，允许你通过点击“more”（更多）链接直接在页面上查看额外的文本。页面的 HTML 代码中并不包含额外的文本，且在首次加载页面时，DOM 中也没有。但在你点击“more”链接，激活脚本后，额外的文本将被添加到 DOM 中的适当位置并显示在页面上。

你可以在 CSS-Tricks 上找到由 Chris Coyier 所写的“What Is the DOM?”（http://css-tricks.com/dom/）一文，其包含了对 DOM 更详细的解释。

11.3.8 渲染<head>元素

一旦 DOM 准备就绪，浏览器开始渲染 HTML 文档。它从 `<head>` 中的第一个元素开始，逐一处理每个元素。

1. 外部资源

当浏览器遇到链接外部文件（如 CSS 或 JavaScript 文件）的元素时，它会加载该文件。

2. 并行加载

每个外部 CSS 或 JavaScript 文件需要一个单独的 HTTP 请求（也就是一个对服务器的请求）。

尽管老版本浏览器一次只能下载一个文件，但新的浏览器可以同时下载多个资源。这使得这个过程会进行得快一些，然而它一次也只能加载几个文件，如果有很多文件的话并不能立即完成加载。

如果浏览器缓存了资源，它们可能根本不需要重新下载。大多数时候，当你加载一个网页时，浏览器将缓存（或在本地保存）资源，比如图像、CSS 文件和 JavaScript 文件，所以在你浏览同一网站的其他页面或者如果你在设定的时段内又返回到网站时它们都不需要再次下载。这意味着用户在网站中访问的第一个页面通常会比后续页面花费更长的时间来加载，因为之前还没有资源被缓存。

文件可被缓存的时间是在网站服务器上设置的，时间范围可以是 24 小时到一年不等。

3. 单线程执行

JavaScript 是单线程的，这意味着浏览器一次只能执行（运行）一个文件。

浏览器执行它所遇到的每个 `<script>` 元素（从 `<head>` 中的 `<script>` 开始），这包括所有的内联脚本和外部脚本，它们只有在下载完成之后才能开始执行。

在开始执行 `<head>` 中的脚本之时，用户看到的仍然是空白页面，因为浏览器尚未开始处理 `<body>` 中的 HTML 代码。

11.3.9 渲染<body>元素

在浏览器完成对 <head> 元素的处理之后，它开始渲染 <body> 元素。

浏览器从 <body> 中的第一个元素开始，逐一对元素进行渲染（一次渲染一个）。浏览器需要知道每个元素应是什么样子，将出现在页面的什么位置，因此它会检查 CSS 以决定怎么做。

1. 加载HTML图像

当浏览器遇到 <img> 元素时，它开始加载对应的图像文件。如果图像文件较大，这可能需要一段时间，因此，用户在图像加载完成之前，可能会在页面上该图像所在位置看到空白。

过去常见的做法是在 HTML 标记中指定每个图像的 `height` 和 `width` 属性，为图像“保留”一个正确尺寸的空白区块。但在响应式设计中，图像的大小可能会随视口宽度的不同而改变，所以这种做法不再适合。

为此的代价是，图像只有在加载完成后，浏览器才能知道图像的尺寸，并有可能必须回流页面的部分内容（在布局中移动内容）来为图像腾出空间。

2. 加载背景图像

当浏览器遇到元素设置有背景图像（通过 CSS 应用）时，它会去加载背景图像。

3. 更多的JavaScript

在 <body> 元素中也可以包含 JavaScript，链接外部文件或者作为内联脚本都行。浏览器无法同时渲染页面和执行脚本。所以，如果浏览器在 <body> 中遇到一个 <script> 元素，在执行对应的脚本时它会停止渲染，直到脚本执行完毕才继续渲染页面。

11.3.10 onload事件

当页面中的所有内容都加载和渲染完毕后，将执行 `onload` 事件中的脚本。`onload` 事件是指页面一完成加载即触发的事件。

11.4 性能测量

可以用一些在线工具测试网站的性能，它们会预估你的页面所需要的下载时间。

Akamai Mobitest 公司的 Mobitest（http://mobitest.akamai.com/m/index.cgi）是一个伟大的工具，可以帮助你测算网站的速度。只要输入页面的网址，在几个不同的移动设备中选择一个，然后请求运行测试即可。所有测试请求都在一个队列内进行，因此你需要等上一小会，在测试运行结束后你可以获得页面的平均加载时间（以秒为单位）和平均大小（以 KB 为单位），如图 11-2 所示。

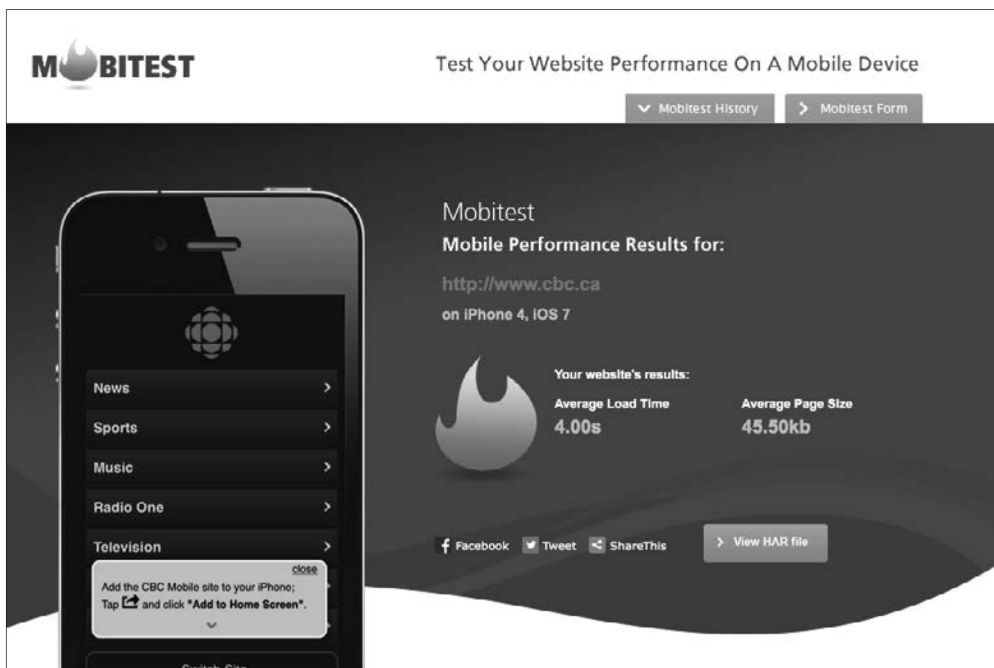


图 11-2：Mobitest 中的速度测试结果

例如，在 iPhone 4 上测试 Yahoo.com 的加载时间是 5.41 秒，页面平均大小是 853.49 KB。在同一部手机上测试 Google.com 的加载时间是 2.9 秒，页面平均大小是 359.47 KB。

你也可以点击“View HAR file”（查看 HAR 文件）来查看显示页面资源加载顺序的瀑布图，以及每个资源加载所花费的时间。这能让你了解是否有哪个特定的页面资源存在问题——例如，一个第三方插件明显拖慢了网站的速度。Mobitest 的工作方式如同一个代理服务器，使用真实的手机测试你的网站，因此它给出的结果是非常准确的。

YSlow（http://developer.yahoo.com/yslow/）是另一个页面性能分析工具，如图 11-3 所示。它是一个浏览器插件（或是桌面 / 移动浏览器书签工具）。当它分析一个网站时，会评估经常出现问题的那些区域，并给出网站优化建议。例如，它可能会指出生成的 HTTP 请求太多了，或者可通过最小化 CSS 文件来节省一些流量。

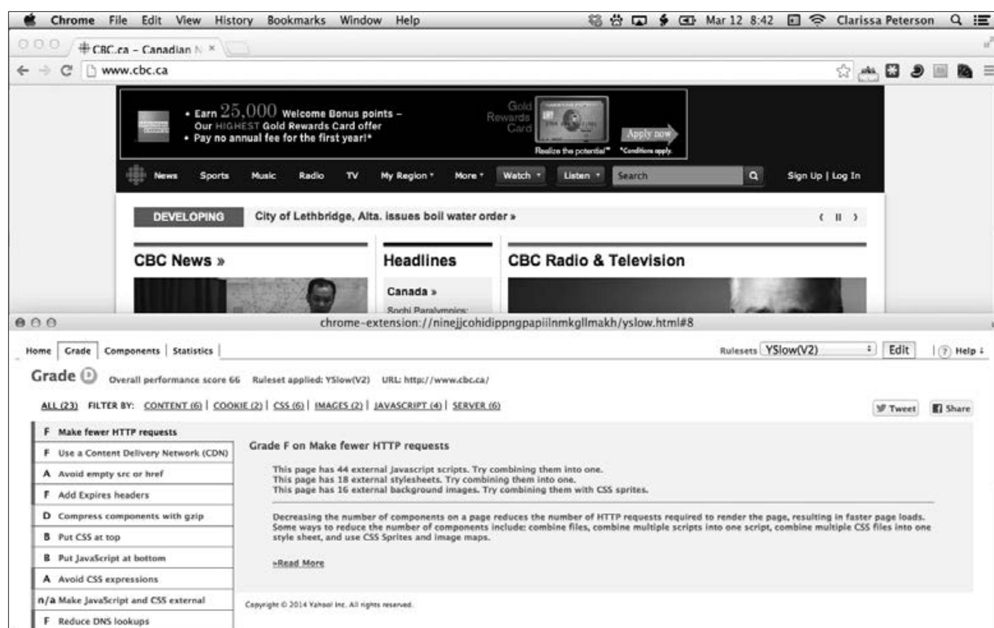


图 11-3：YSLow 显示了需要处理的特定性能区域的评级

其他工具还有 WebPagetest（http://www.webpagetest.org）和 Pingdom Website Speed Test（http://tools.pingdom.com/fpt/）。

11.5 清理代码

改进网站性能首先要做的就是清理代码，使之占用的空间更少。

11.5.1 使用简单直接的代码

编写代码应尽可能简单直接似乎是显而易见的，但请尽你所能。

如果有其他办法就不要只是为了将样式应用于某个元素就为其添加一个 class 或 id 属性。例如，下面这么做是不必要的：

```
<header>
<ul id="navigation">
...
</ul>
</header>
```

如果 `<header>` 中只有一个 `<ul>` 元素，就不需要为其设置 id 属性，可以直接这样应用样式：

```
header ul { ... }
```

这种方式显然易见，但在编码过程中往往不经意间就会为元素添加额外的 class 和 id。直到后来回顾和查看代码时，你才可能意识到它们是多余的。

利用层叠来避免重复的 CSS。例如，如果网站中的绝大多数文本使用同一字体，应将 `font-family` 样式应用于 `< body >` 元素，而不是分别应用于 `<p>`、`< li >` 等元素。然后再对例外的元素追加特定的 CSS 样式。

不要忘记回顾并清理你的样式表，因为可能有的样式已经失去作用（针对的元素不存在了）。

虽然在写代码的时候最好做到简单直接，但有时很难预见什么会变得必要或不必要，所以要确保在最后对代码进行回顾来找出可以去掉的代码片段。

11.5.2 缩小

我们在编写代码时会添加很多的空白，因为这能使代码更容易阅读，但是浏览器并不需要这些额外的空白（空格、换行和缩进字符）。

缩小文件就是去除文件中这些不必要字符的过程。例如，像下面的代码：

```
p {
    font-family: Georgia, serif;
    font-size: 1.1em;
}
li {
    font-family: Helvetica, sans-serif;
    font-size: 1.2em;
}
```

对浏览器来说只需要这样即可：

```
p{font-family:Georgia,serif;font-size:1.1em;}li{fontfamily:
Helvetica,sans-serif;font-size:1.2em;}
```

在上述例子中，我们能够去除 10 个空格符，2 个制表符和 7 个换行符——将近 15% 的字符！

当然，在编写 CSS 代码时像这样不用空白字符是难以阅读和维护的，并且在一个文档中不能简单地搜索和替换掉所有的空格和换行符，它们中的一些是必不可少的（比如，如果把 `nav li` 中的空格去掉变成 `navli`，这就会导致原本的样式不起作用）。

因此，我们可以在上传到网站服务器之前，再将 CSS、JavaScript 和 HTML 文件用“缩小工具”缩小，同时还可像去除空白字符那样去除掉注释（对浏览器没必要）。

有以下几种方法可以实现。

首先，对于非常小的不常编辑的网站，可以在上传到网站服务器之前单独缩小每个文件。如果是一次缩小一个文件，有几个网站可通过简单的形式来执行此操作，比如 CSS

Minifier（http://cssminifier.com）或 YUI Compressor（http://refresh-sf.com/yui/），或在网上搜索“CSS minifier”或“JavaScript minifier”。但对于大多数网站，手工缩小文件要做太多额外的工作（不止一个文件）。

如果网站只有一个工作人员，那也简单，假如所有的文件都保存在自己的电脑上，可以使用像 LiveReload（http://livereload.com）或 Mixture（http://mixture.io）这样的预处理器应用程序在上传文件到服务器之前缩小它们，而在自己的电脑上总是可以编辑带有空白字符和注释的原始文件。

如果是与他人共用这些文件，要稍微麻烦些，因为每个人都必须使用相同的原始文件来工作，而不是从服务器下载处理过的最新版本的文件。

通过使用像 Minify 这样的工具（用 PHP 编写），你可以设置在运行时服务器自动缩小文件，但额外的处理时间往往抵消了发送更小文件所带来的好处，所以这种方法并不推荐。

11.6　将HTTP请求减至最少

每当浏览器请求服务器提供一个文件时都会产生一个 HTTP 请求。

每一个文件，比如图像、样式表或网络字体，都需要一个单独的 HTTP 请求。每一个文件从浏览器到服务器再返回这样一个“往返”的请求 / 响应过程在宽带上可能只需要零点几秒的时间，但在移动网络上则可能需要整整一秒的时间。

提高网站性能的一个最有效方法是将对服务器的 HTTP 请求数量减至最少。可以通过合并 CSS、JavaScript 或图片文件来实现此目标。

还需要确保网站中嵌入的第三方代码（比如社交媒体微件、广告或者统计分析脚本）不会请求大量的文件。

11.6.1　串联

减少 HTTP 请求的一个简易方法是通过串联（concatenation）合并 CSS 和 JavaScript 文件，从而减少所调用的 CSS 和 JavaScript 文件的数量。

虽然对于开发人员来说，根据用途的不同将代码分拆到单独的文件中可能更方便，但由于每个文件都需要一个单独的 HTTP 请求，这将延长页面的加载时间。

如果出于开发方便的目的有多个分拆的文件，那么可在上传文件至服务器之前再将多个文件组合成一个或两个文件。许多用来缩小文件的应用程序也可以用来串联文件，比如 YUI Compressor（http://yui.github.io/yuicompressor/） 或 Minify（https://code.google.com/p/minify/）。也可以使用 QuickConcat（https://github.com/filamentgroup/quickconcat），它是一

个用 PHP 编写的运行于服务器端的工具。

显而易见，某些文件是不能被合并的，比如，如果在样式表链接中使用媒体查询对不同的屏幕尺寸指定了不同的样式表，或者是某些仅用在网站部分页面上的样式表。

此外，也很容易通过合并代码到几个文件而不是分散在一堆文件中来单纯减少样式表的数目。对于合并后的样式表你可以通过注释分隔各部分的样式以便于引用，同时使用缩小工具删除线上版本样式表中的注释。

不过，不要走极端。浏览器具有一次同时并行下载几个文件的能力，所以有三两个 CSS 文件可能要好过虽只有一个 CSS 文件但文件却非常长这种情况。

11.6.2 第三方代码

任何从其他域名下载的资源都会导致延迟。这可能包括嵌入式视频、地图、广告或统计分析脚本。

主要的问题是，在页面加载过程期间，每一个额外的域名都会增加额外的 DNS 轮询。如果所有的内容都是来自同一个域名则可避免此问题。

第二个问题是，你提供的内容依赖于别人的网站。如果别人的网站变慢或不可用，你的网站在试图加载资源时就会因此受阻而无法继续或被卡住。

最常见的罪魁祸首之一是社交媒体微件——你肯定见过，就是网页中的那些 Facebook、Twitter 或其他网站的“Share This”（分享）链接小图标。

这些往往是使用第三方软件在页面中实现的，允许你跟踪用户的行为，问题在于它们通常由大量代码组成。

图 11-4 展示了《华盛顿邮报》网站上的社交媒体微件。



图 11-4：《华盛顿邮报》网站上的社交媒体微件

它看上去够简单的，但部分问题在于底部的“更多”（More）链接，其包含其他社交媒体网站的链接。这些底层代码并非最优——每个选项由一个 `<div>`、`<li>` 和 `<span>` 元素组成，连带数个 CSS 类和一个 `onClick` 事件。

更不用说所有的图标了，即使是合并在一个拼合图片中，它们仍然有 22 KB 大。

你应考虑是否真的需要这么一个复杂的微件，即使没有它用户仍然可以分享你的内容。Oliver Reichenstein 在 Information Architects Inc 网站上发表了一篇伟大的博客文章“Sweep the Sleaze”（扫除劣品），细数了社交媒体微件并不能增加网站价值的原因（http://ia.net/blog/sweepthe-sleaze/）。

但是如果你确实需要保留这样的通常由 JavaScript 生成的微件，请仔细考虑下是否可以推迟到页面渲染完毕之后再加载它们（在本章随后的部分，我们将进一步讨论推迟 JavaScript 脚本执行的话题）。当然推迟加载后，它会延迟一会才在页面上弹出来，但这并没关系，因为通常用户是在已阅读了页面之后才会决定是否要将其分享给他人。

[小贴士]

如果你真的需要社交媒体微件，可以研究下 Filament Group 公司的 SocialCount（http://filamentgroup.com/lab/socialcount.html），它是一个 jQuery 插件，允许你以一种不会严重影响性能的方式添加社交媒体微件，插件包含的 JavaScript 和 CSS 加起来只有 3 KB。

11.6.3　图像拼合

我们已经在 6.1.4 节详细讨论过图像拼合，它是一种通过将几个小图像组合成一个大图像进而减少要下载的文件数目的方法。

当图像彼此相关时这种方法是最有效的（比如，一组显示在一起的图标）。

11.7　服务器设置

接下来我们将学习必要的服务器设置。如果你是一个设计师，它们可能不在你的直接掌控之下，而那些有控制权的人可能又认识不到这些设置对网站性能的重要性。但这并不意味着你就可以无所作为，找到那个有控制权的人并促使其合理调整设置是绝对值得的。

11.7.1　避免重定向

HTTP 重定向发生在用户所加载页面的网址不是页面的实际网址之时，浏览器被重定向到正确的网址。这需要在网站服务器上进行设置，比如通过设置 Apache 的 .htaccess 文件，或者通过设置内容管理系统（CMS）中的功能选项。

最熟悉的重定向是从备用域名访问一个网站。例如，如果在浏览器中输入 www.newyorktimes.com，页面加载后，你会看到在地址栏中实际的网址是 www.nytimes.com。

当页面的网址发生了变化而你希望确保访问旧网址的用户仍能获得正确的页面，或者你想将用户导向网站的一个特定子域名。这也是我们经常使用重定向的两种情形。

在有移动、桌面两个版本的网站上，通常会基于用户使用的设备将用户重定向至正确版本的页面。例如，从 www.example.com/pagename 重定向至 m.example.com/pagename。（对于响应式设计，你不必担心这种类型的重定向。）

很明显，有很多场合需要使用 HTTP 重定向。但请记住，重定向是有性能损失的。

尤其要确保所有网站使用的文件，包括图片或样式表，使用了正确的网址调用。比如，在链接同一个域名下的文件时应使用相对链接（/images/file.jpg）而不是绝对链接（http://www.example.com/images/file.jpg）。

如果网站中页面的地址在重新设计网站后发生了变化，只要有可能就应该在网站上启用实际的链接，而不是使用重定向将用户导向正确的位置。

更应避免多个重定向连在一串这种情况。

11.7.2　文件压缩

压缩可用于减小所有要发送文件的大小。

大多数网站服务器可以在发送文件给用户电脑之前使用 gzip 格式压缩文件。当浏览器请求一个页面时，会告诉服务器它是否能处理压缩文件。服务器将向能够处理压缩文件的浏览器发送压缩过的文件。

可以使用 GIDZIPTest（http://www.gidnetwork.com/tools/gzip-test.php）来检测一个网页是否是压缩过的，如果是的话，则会给出压缩前后文件大小的比较。例如，测试《华盛顿邮报》网站的首页，能看到它是用 gzip 格式压缩过的，节省了 69.5% 的空间（300 KB 压缩成 92 KB)。

可以在网站服务器设置中打开压缩功能。例如，Apache 站点使用 `mod_deflate` 模块提供压缩功能（http://httpd.apache.org/docs/current/mod/mod_deflate.html）。你的虚拟主机提供商可能会为此在控制面板上提供一个设置选项，并且它可能是默认启用的。如果是在一个共享主机上，则可能无法打开和关闭压缩功能。

我们应该对所有的 HTML、CSS 和 JavaScript 文件都进行压缩。

11.7.3 浏览器缓存

浏览器在显示一个网页时会下载大量文件，这些文件将在用户浏览网站其他页面时被重用，其中包含了样式表、图片、JavaScript 文件和字体文件。

*浏览器缓存*意味着浏览器可以在指定的时间段内将这些文件临时存储在用户的电脑上，这样一来，当用户访问网站的其他页面时，或者如果她之后某天又回来访问网站时，浏览器不需要重新加载这些文件。

通过减少需要下载的数据量不仅可以减少页面的加载时间，还可以减少下载页面所需的带宽，这对于带宽有限的手机合约计划用户或按流量付费的手机用户都是有益的。

浏览器缓存并不必然是自动产生的。要去缓存什么并不由浏览器决定，相反，是由网站服务器设置的一组规则来告诉浏览器是否以及何时去缓存某一文件。

如何设置浏览器缓存取决于所使用的网站服务器软件。例如，在 Apache 中，应编辑 .htaccess 文件。

你的主机可能默认打开了浏览器缓存。如果不确定的话，可以使用谷歌的 PageSpeed Insights（http://developers.google.com/speed/pagespeed/insights/）来检查一下。

我们可以为不同类型的文件设置不同的过期时间。例如，可以指定 .jpg 文件在一个月后过期。之后浏览器会缓存所有的 .jpg 文件，而不会每次都尝试重新加载它们，直到文件在首次下载一个月后过期。

文件的缓存时间取决于文件的类型以及文件的更新频率。例如，你可能不经常改变网站上的图片文件，因此允许长时间缓存它们是安全的。如果需要改变一个图片文件，比如更换新版网站标志，可以简单地给替换图片一个不同的文件名，那么浏览器将被迫下载它。

如果更改样式表非常频繁，你可能需要设置它们的过期时间为一天。这对用户的好处还是他们在浏览网站时不必每次都重新下载这些文件，但对于你所做出的任何样式变化他们将在 24 小时之内才能看到。

请记住，浏览器缓存空间可能是有限的，所以如果一个用户访问大量的网站，那么并不是所有具有长过期时间的资源都能实际存储那么长的时间。这在老式的移动设备上尤其是个问题，它们拥有的缓存空间非常有限。但即使只有极小的缓存空间，在用户此次访问网站期间，文件将保留在缓存中，除了一开始访问第一个页面外，其他页面的加载时间都会因此而变短。

一般来说，静态资源应该至少设置一周的过期时间，可能的话还可以更长。缓存时间最长允许设置为一年。只有那些一天改变多次的资源，如新闻网站的首页，应该设置成根本不缓存。对于经常改变的资源即使只允许浏览器缓存几个小时也能减少网站的加载时间。

11.8 JavaScript

如果不熟悉 JavaScript，你需要了解一下它的工作方式。下面的 11.8.1 节和 11.8.2 节提供了一点儿背景知识。如果你已经熟悉 JavaScript，完全可以跳到 11.8.3 节。

11.8.1 JavaScript做什么

你大概已经知道了网站是分层构建的，第一层是用于构建内容的 HTML，第二层是用于呈现内容的 CSS。

网站的第三层是 JavaScript，它用于交互或行为。这是一个可选的层，如果只是要求创建一个供用户阅读和观看，只具基本交互性的网站，你可能根本不需要用 JavaScript 就能创建出一个好的样品来。

JavaScript 常用于验证表单字段，更新页面部分内容的同事不必重新加载整个页面，以及页面元素的动画效果实现。

JavaScript 是一种客户端编程语言。这意味着所有的 JavaScript 代码是完全由浏览器（客户端）加载并执行的。这与 PHP 之类的服务器端语言形成对照，后者在服务器端完成对页面的处理后再将页面发送给浏览器。

11.8.2 JavaScript的工作方式

JavaScript 可以包含在 HTML 页面中，或者单独保存为扩展名为 .js 的文件（通过 `<head>` 或 `<body>` 中的 `< script >` 元素链接）。

浏览器加载 HTML 文档之后，它从 `<head>` 第一个元素开始，顺序加载、呈现、执行它所遇到的每一个元素。所以，假如 `<head>` 中的第一个元素是外部 CSS 文件的链接，它将先加载这个 CSS 文件然后才移动至下一个元素。

当浏览器遇到 JavaScript 时，不管是链接脚本还是内联脚本，它都会加载并执行完脚本之后才移至下一个元素。

所以，如果你在 `<head>` 中包含有脚本——不管是采用内联形式还是通过链接外部 JavaScript 脚本文件，浏览器会在处理 `<body>` 元素，开始向屏幕显示内容之前先执行这些脚本！这意味着在脚本运行期间，用户将盯着一个空白的屏幕。

不过，这并不意味着在当前 JavaScript 执行时，任何效果都能够实际生成。

大多数 JavaScript 是由一个事件触发的。例如，你可能见过 `onclick` 事件，其意味着在用户点击指定的元素之前，不管 JavaScript 要干什么，都不会发生。`onload` 事件则意味着仅

在页面加载完成之后事件中 JavaScript 才开始执行。另一个例子是在用户点击提交按钮时执行表单字段验证的脚本。

网页中的很多 JavaScript 脚本只是为了之后能通过类似 `onclick` 这样的事件引发将执行的操作而做准备的。

不幸的是，因为取决于你在页面的什么位置加入或链接 JavaScript 脚本，这些脚本可能会导致很多问题，可能会阻塞页面的渲染，大大减慢渲染的速度。

此外，浏览器的工作方式意味着浏览器不能同时渲染页面和执行脚本。因此尽管从某种意义上说，浏览器一次不只是能做一件事情（这个解释起来很复杂），但如果浏览器遇到一个脚本，在它执行脚本期间它必须停下所有的渲染工作。

11.8.3 阻塞式JavaScript

阻塞页面加载的 JavaScript 有时称为*阻塞式* JavaScript。

因为浏览器在遇到脚本时必须运行它，浏览器可能在渲染 HTML 和 CSS 之前被迫运行所有 JavaScript，即使它在页面完成加载之前，并不需要这其中的大多数 JavaScript。

如果不确定你的 JavaScript 是否会阻塞页面的渲染，试试谷歌 PageSpeed Insights（http://developers.google.com/speed/pagespeed/insights/），它将告之网站是否存在阻塞元素。如图 11-5 所示。

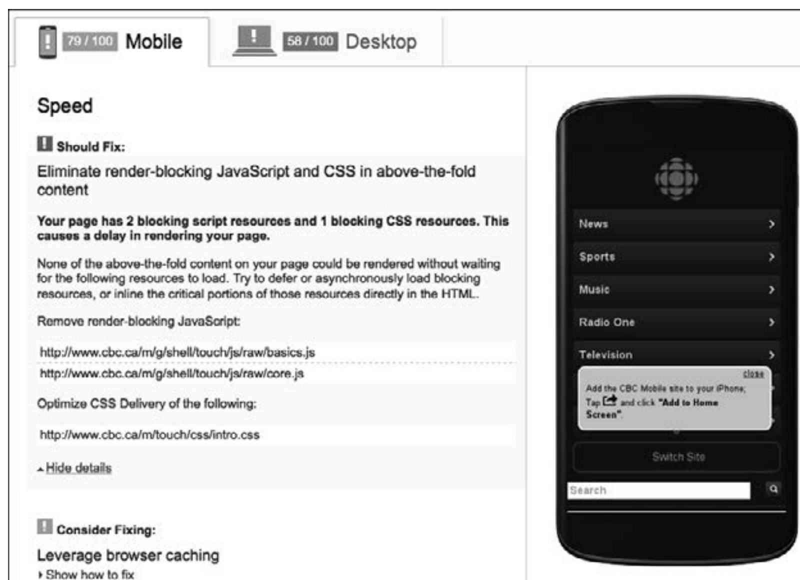


图 11-5：PageSpeed Insights 会给出网页的 JavaScript 是否阻塞了页面的渲染

要防止 JavaScript 阻塞页面的渲染，则应该确保在浏览器加载和渲染页面时，只加载必要的 JavaScript 来显示页面上用户首先会看到的内容。所有其他的 JavaScript 应该在页面渲染完成之后再加载。

1. 优先加载首屏所需JavaScript

如果 JavaScript 对于显示首屏（above the fold，一眼就能看到不用向下滚动屏幕）页面内容是必需的，那该脚本应该在加载页面时即被加载。

其他的 JavaScript 可以推迟加载（我也将这个问题推后到下一节讲述）。这将包括所有只在页面加载完成之后才需要的脚本（比如 `onclick` 事件处理器）或应用于页面很靠后内容的脚本。

仔细考虑每个脚本是否真的立即需要，以及是否可以推迟执行。

对于响应式网站，你可能会用到 JavaScript 腻子脚本或特性测试来使网站能在老式浏览器上工作（即确保媒体查询能在老版本 IE 上工作）。这些脚本需要放在 `<head>` 元素中，因为它们对于在那些老式浏览器中渲染页面是必需的。

2. 内联脚本

如果你有小段的脚本，那么可以以嵌入形式将它们包含在 HTML 中，这样浏览器就不必加载额外的文件了。

潜在的问题是缓存的限制。如果你包含了内联脚本，每次页面加载时它也随页面一起加载。所以这对于大型脚本或者网站中所有页面都需要的脚本可能不是一个好主意，但如果网站的首页有些特殊的内容需要 JavaScript，使用内联脚本可能是个不错选择 :

```
<head>
    <script type="text/javascript">
        ...
    </script>
    ...
</head>
```

如果网站只用到一点点 JavaScript，通常最好以内联方式包含它，而不是添加一个需要另外下载的文件。

3. 延迟加载

对于页面晚些时候才需要的其他脚本，你可以通过把它们移到页面尾部 `</body>` 标记之前，将其延迟至页面渲染完成之后才加载。

这些脚本将不再阻塞页面的渲染。只有页面上的最后一个 HTML 元素渲染完之后这些脚本才开始加载并执行。

4. 推迟执行

有点棘手的是使用了 defer 属性的 <script> 元素。如果你在 <head> 中使用这样一个 JavaScript 链接，浏览器在首次遇到该元素时仍会加载链接的脚本文件，但会推迟文件的执行直至网页渲染完毕之后。

该属性是一个布尔属性，这意味着如果 defer 属性存在则表明允许推迟执行，如果该属性不存在，则表明不允许推迟执行：

```
<script type="text/javascript" defer>
    ...
</script>
```

defer 属性对所有当前的主流浏览器都有效，除了 Opera Mini 浏览器。查看文章“Can I use defer attribute for external scripts？”（http://caniuse.com/script-defer）以了解更多细节。

注意，这只对外部脚本（到 JavaScript 文件的链接）有效，并不适用于内联脚本。

一般来说，最好就是将 JavaScript 移至 HTML 文档的末尾。如果不是计划在页面渲染之前执行 JavaScript，那么在页面渲染之前加载它就毫无意义。

5. 异步加载

HTML5 中新增加了一个 async（即 asynchronous，意为“异步”）属性。

如果 <script> 元素包含有该属性，当浏览器在 <head> 或 <body> 中首次遇到它时，浏览器会一边继续解析页面一边开始加载 JavaScript 文件。这与以往的不同之处在于，在浏览器加载并执行那个特定的脚本时，它不会停止其他的加载和渲染工作。

相反，它会在加载其他资源（如果有的话）的同时加载脚本，并在脚本一旦可用时（即当脚本完成加载后）执行它。

注意，这个属性只对外部脚本（到 JavaScript 文件的链接）有效，并不适用于内联脚本。

该属性也是一个布尔属性，因此如果 async 属性存在则表明允许异步加载，如果该属性不存在，则表明不允许异步加载：

```
<script async src="example.js">
    ...
</script>
```

这个属性被除 Opera Mini 之外的所有最新版本的浏览器所支持。

通常，如果一个脚本是渲染页面所必需的，应把它放在 <head> 元素中，并考虑包含 async 属性，只有一个例外：如果有多个脚本需要按特定的顺序（例如一个依赖于另一个）来执行，则不要包含这个属性。每个包含 async 属性的脚本只要加载完成就会被执

行，但它们完成加载的次序可能并不与你在 `<head>` 中放置它们的顺序相同，因此它们可能也不是按那个顺序来执行的。

当你从第三方来源加载 JavaScript 文件时（比如当你有嵌入的地图或广告类的内容时），你可能特别需要使用异步加载，这样，如果从其他网站加载文件时遇到问题，它并不会阻塞网站上其他内容的加载。

6. 仅加载页面必需的代码

额外一点，你应该确保在每个页面中只加载必要的 JavaScript。

在 `<head>` 中包含所有网站要用到的脚本文件，这一做法仍然非常普遍，即使某些文件只是用在首页上，或只是用在站点的部分页面（比如电商页面）上。

尽管文件缓存意味着脚本文件不会反复加载，但在用户访问网站时，他所访问的第一个页面相对而言会慢很多，因为需要去下载所有的脚本文件。

7. 用HTML/CSS代替JavaScript

你可能并不知道，可以纯粹用 HTML5 和 CSS3 做很多原本你习惯于用 JavaScript 来实现的事情。

例如，如果表单包含一个日期表单字段，以前需要使用 JavaScript 来创建一个小的弹出日历框让用户可以选择日期。在 HTML5 中，引入了新的日期类型表单字段，它会基于浏览器的默认样式自动创建一个日历，而不需要任何 JavaScript（尽管这还不被所有浏览器支持）。

你也可以使用 CSS 实现当鼠标悬停于一幅图片上时将其变成另一幅图片的动态效果。

在大多数情况下，使用 HTML 或 CSS 代替 JavaScript 将有助于提升性能。浏览器不必加载大量的代码（创建和样式化日历弹出框的代码是非常复杂的），如果浏览器已经支持新的日期型表单字段，你只需使用 `<input type="date">` 即可。

不过，HTML 和 CSS 并不总是一个更好的选择。例如，你可以使用纯 CSS 做一些非常复杂的动画，但有时用 JavaScript 实现相同的效果对性能的影响会更小。你最好对它们进行单独考察，从中挑选出最佳的方法。

为了用 JavaScript 实现一些花哨的效果而去牺牲性能根本不值得。记住，性能也是设计的一部分。

11.8.4 JavaScript库

使用 JavaScript 库或框架是使网站膨胀的另一个原因。

一个 JavaScript 库是一组预先写好的 JavaScript 代码，便于人们编写网站所用的脚本。基本

上，它包含一些在网站中使用的常用功能，你只需要敲入很少的 JavaScript，而不必从头编写所有的 JavaScript 代码。

有许多不同的 JavaScript 库，最著名的是 JQuery（http://jquery.com）。JavaScript 框架类似于 JavaScript 库，只是具有更多的功能。

向网站添加 JavaScript 库所带来的问题是，你可能只会用到它的一小部分功能，却被迫接受库中所有你并不需要的其他代码。

如果要在网站中使用一个 JavaScript 库，查看下你是否真的需要一个完整的库。现在可以找到很多微库或微框架，它们要小得多（一般不超过 5 KB）并只包含执行某个特定任务的代码。请访问 Microjs（http://microjs.com）网站，其中收集了一系列数以百计的小脚本，并可对脚本进行精确地搜索。

当然，你也可以自己编写 JavaScript 脚本，而不去依赖别人预先写好的代码。

11.9　CSS

正如前面提到的，使用 CSS 的第一步是最小化你需要的 CSS 样式数量，使用简单直接的代码，并且避免重复不必要的样式。

CSS 也可能阻塞页面的渲染。浏览器在等待所有的 CSS 文件加载完毕之后才开始渲染页面。因为它需要根据 CSS 来获知每个元素在何处以及用怎样的方式来渲染。

使用 `<link>` 而不是 `@import` 可以加速样式表加载，前者使得浏览器可在同一时间加载数个样式表，后者则使样式表按顺序逐一加载。

你可做的另一件事情就是延迟加载那些不是初始页面（首屏页面）所必需的 CSS，将它们移至页面尾部 `</body>` 标记之前，就像你对 JavaScript 所做的那样。

如果有很多仅适用于网站某些特定部分的样式规则，将这些样式放在一个单独的样式表中，仅在需要的页面上加载它们。

CSS框架

使用 CSS 框架会导致使用 JavaScript 库类似的问题，最终会有很多实际上并不需要的代码。一个框架涵盖了网站可能会做的所有事情，并包括所有的实现代码。因此，对于使用它的大多数网站实际用到的功能而言，框架显示过于复杂。

如果你使用一个类似 Bootstrap 或 Foundation 这样的框架来构建你的网站，请注意这个问题。

框架中的 CSS 通常是容易理解的，因此你应该能够梳理和排除大部分你知道自己不会用到的 CSS。

例如，如果网站不包含任何数据表格，就可以移除所有关于表格的 CSS 样式。当然，如果以后又要在网站中使用表格的话，总是可以将这些 CSS 再添加回去。

虽然这么做会多花点时间，但通过梳理和移除不会在网站中使用到的 CSS，可以节省更多的代码空间。

[提示]

将 CSS 框架中所有未使用的 CSS 移到一个名为 unused.css 的文件，并把它与其他 CSS 文件一道放在服务器上，这样在你需要时，就能很方便地将移除的代码添加回去，不用去管这个文件有多大，因为它不会随网站的其他资源一起被下载。

此外，确保你没有重复编写 CSS 规则。CSS 框架几乎为所有的元素都设置了 CSS 属性值。如果你需要改变什么，比如 `<h1>` 的字体大小，不要简单地在文件末尾添上新的 CSS。相反，应在框架的样式表中找到 `<h1>` 并替换原来的值。在文件末尾添加一个单独的声明不仅仅是增加了不必要的代码，也可能给后来试图编辑样式表的开发人员造成困惑。

最后，尽管 CSS 框架可能用起来非常方便，比如当你需要快速搭建起一个网站，或者你雇用来创建网站的员工没有丰富的开发经验（比如非营利组织、小企业或者业余爱好项目），但你得到的网站肯定要比从零开始完全自主编写代码所创建的网站要臃肿得多。

11.10 托管

将网站托管在不同的地方肯定会对加载速度造成影响。

如果每月只花几美元托管在共享主机上，网站肯定要比你花大价钱购买独立服务器及更好的数据连接要慢。不过更贵并不意味着一定就更好。货比三家，并将各家主机服务商的客户满意度作为参考，再做出选择。

另外，如果托管方案有带宽限制，在超过流量限制后，服务商可能会限制网站的速度（或完全禁止你的网站）。

11.10.1 内容分发网络

可以使用内容分发网络（Content Delivery Network，CDN）作为网站托管解决方案的一部分。当使用 CDN 时，网站的所有文件不是都驻留在一台服务器上，而是多个副本文件被复制到不同地点的数据中心的多台服务器上。当用户向网站请求内容时，CDN 将根据地理

位置、当前网络流量模式等因素将请求分发至最优的服务器。

这有助于提升任何站点的性能，虽然说它对实时、流媒体或大流量网站尤其管用。

一个网站可以选择只托管部分内容在 CDN 上。例如，一个大流量的电子商务网站可以将其所有的图片托管在 CDN 上（因为网站有非常多的大图片），而将其他内容托管在普通的主机上。

你可以在购买网站主机时附带购买 CDN 服务，或者单独从某些公司购买（通常是互联网服务提供商）。对于非常小的网站价格可能只要 10 美元 / 月，而大型网站所需的费用可能高达每月数千美元。

11.10.2　内容管理系统

CMS（内容管理系统）也可能拖慢你的网站。企业级 CMS 比简单的个人 CMS（比如 WordPress 或 Drupal）问题更大。

CMS 响应浏览器发送网页的请求时，需要做很多的工作，而不仅仅如发送一个纯 HTML 页面那么简单。你的整个网站数据都包含在一个数据库中，CMS 将运行一堆的查询来实时拼装出页面上所有独立的元素——导航、内容、插件。

理想状况下，这会很快完成，但很多事情可能出错并增加延时。

研究表明在不同的 CMS 之间存在明显的性能差异，所以在 CMS 选型时应研究对比它们的性能。

如果你已经有一个 CMS，检查所有的设置以确保你已经打开了所有可以提高性能的选项。

始终确保你所用的 CMS 及其插件为最新的版本，过时的软件运行速度往往较慢，并可能导致冲突，因而增加延时。

11.11　有条件地加载内容

前面已多次提到过，我们要尽力确保相同的内容可用于所有用户，而不管用户使用的是什么设备。但是这并不意味着在同一时间的同一个页面上所有的内容都要是一览无遗的。

例如，对于正文，可能会有补充内容，在宽屏视图中能很容易地将其安放在侧边栏中。但在窄屏视图上，你不想占用一大块屏幕空间来显示该内容，但你仍会给用户一个查看它的选项。

这就是*条件加载*的用武之地。这种技术使用 JavaScript 查询浏览器窗口的宽度（类似于一个 CSS 媒体查询），然后仅在大于最小尺寸的屏幕上运行一个函数来显示补充内容。

> **[小贴士]**
>
> 用户能够接受网站依据屏幕的大小显示不同的内容，只要他们能够用另一种方式访问那些被隐藏的内容，在本案例中，小屏幕的用户只需点击链接即可查看补充内容。

Jeremy Keith 在 24 Ways 上阐述了这种技术“Conditional Loading for Responsive Designs”（http://24ways.org/2011/conditional-loading-for-responsive-designs/）。

方便起见，我们直接使用 Keith 文章中的例子，假设你想要以列表形式提供谷歌新闻中与“猫”相关的最新的新闻报道。

在小屏幕上，你不想包含所有的新闻，那你可以就只放一个链接，它指向相关话题的谷歌新闻搜索结果页面：

```
<div id="newsresults">
    <a href="http://www.google.com/
    search?q=cats&tbm=nws">Search Google News for Cats
    </a>
</div>
```

然后在页尾添加一个 JavaScript 函数，该函数将在页面宽度足够时用谷歌搜索结果替换 `id` 属性为 `newsresults` 的 `<div>` 元素中的内容（如果想看看实现代码，请查看 Keith 的文章）：

```
<script>
if (document.documentElement.clientWidth > 640) {
    searchNews( ‘cats’ );
}
</script>
```

如果视口宽于 640 像素，将会运行 `searchNews` 函数并改变 `id` 属性为 `newsresults` 的 `<div>` 元素去显示完全不同的 HTML 内容——新闻列表，包含标题、链接和描述，如图 11-6 所示。



图 11-6：条件加载的一个例子

你需要记住的是，条件加载只能在可运行 JavaScript 的浏览器中工作。但这也算不上是一个大问题，因为那些没有 JavaScript 的用户会获得默认的（小屏幕）内容，比如用一个链接代替了新闻列表。他们实际上并不会丢失任何东西。

另外需要注意的是，不像 CSS，只要条件发生变化其就会自动重新渲染页面，JavaScript 函数将只运行一次。所以如果视口的宽度发生改变，比如设备从竖排方向转为横排方向，布局将不会像 CSS 那样能凭借媒体查询自动重新适配新的视口尺寸。

当然，使用这个功能意味着小屏幕用户将加载一堆他们不需要的 JavaScript。但是如果能避免他们加载大量不需要出现在页面上的内容，这是值得的。

11.12 重绘与回流

即使浏览器完全渲染完毕页面，仍会有很多事情可能发生。

有时一个元素的视觉外观在页面加载完后可能会改变。比如当鼠标光标悬停在链接上时其颜色会产生变化。当浏览器需要做出这样的改变时，称为*重绘*。

其他时候，页面的布局可能会发生变化。例如，当用户改变浏览器窗口大小时（或将设备从竖排方向转成横排方向时），弹性元素将会改变其大小及内容，比如文本将会移动位置以适应这一变化。页面布局的改变称为*回流*（reflow）。

当我们在响应式设计或移动版网站这个限定条件下谈论“性能”时，我们通常指的是加载页面所花去的总的时间。但在页面加载完毕后其外观的后续改变也可能或快或或慢，这取决于代码是如何编写的。

上面举的例子都很简单，用户会立即感知到发生了变化。但还有许多其他事情可能导致重绘和回流，比如当页面上有复杂的 CSS 动画时，或者使用 JavaScript 操纵页面时，并且在这种改变完成之前可能会有明显的延迟。

这种情况并非是响应式网站所特有，它本质上是技术问题，超出了本书的范围。但在为移动设备优先设计的情境下，你需要意识到，移动设备的性能往往不如台式电脑，因此重绘和回流可能会明显变慢并会产生你在台式电脑上开发网站时所遇不到的问题。

想了解更多有关该主题的信息，参考 Nicole Sullivan 的博客文章“Reflows & Repaints: CSS Performance Making Your JavaScript Slow?”（http://www.stubbornella.org/content/2009/03/27/reflows-repaintscss-performance-making-your-javascript-slow/）。

11.13 RESS

RESS（responsive design + server-side component）*表示响应式设计加服务器端组件*，人们对此存在一定的争议。RESS 的想法是，你可以使用响应式设计制作一个网站，但同时也使用服务器端解决方案来只发送特定设备所需的代码和文件。

因此，尽管对于一个网站你只有一套代码，且代码允许网站响应所有的视口宽度，但网

站的某些部分仍可以基于浏览设备的不同而有不同的实现。Luke Wroblewski 在 2011 所写的文章“RESS: Responsive Design + Server-Side Component”中详细描述了 RESS（http://www.lukew.com/ff/entry.asp?1392）。

要理解 RESS 是如何工作的，可以想象一个响应式网站，在小屏幕上有一个下拉导航菜单，在大屏幕上是一个横向导航栏。两者的 HTML 代码是相同的，但媒体查询允许你根据屏幕尺寸应用不同的 CSS，使得它在不同的屏幕上看上去是不一样的。

使用 RESS，服务器将检查设备是否是移动设备，并对移动和非移动设备发送不同的导航 HTML/CSS 代码段。这样，每台设备只得到它需要的代码，与之相对，在响应式设计中，每个设备都要下载所有的代码，不管它是否需要。

RESS 需要使用后台编程语言（比如 PHP），而不仅仅是 HTML/CSS/JavaScript，因此它要比基本的响应式设计更难一些。

除了检测是否是移动设备在请求页面，服务器还可以检测设备特性，并基于此发送不同的代码。你也可以用 JavaScript 库 Modernizr（http://modernizr.com）来检测设备特性。

RESS示例

圣母大学网站除采用响应式设计外，还使用 RESS 来为桌面和移动用户分别提供不同的内容。

桌面版网站中各主要部分（关于、学术水平、招生等）全都在一个页面上。在图 11-7 中，左边部分是桌面网站的上半段页面，招生部分还处在箭头所示的更下方的页面中，如果调整桌面浏览器宽度至 320 像素并载入网站，如图的中间部分，你仍能看到所有的内容，只不过所有内容被媒体查询重排成一列。图的右边部分则显示了网站在智能手机中加载后的样子——那就是手机中能看到全部内容。

在手机上，你仍能访问到主要的内容部分，比如招生，但现在它们都变成了一个单独的页面，而不是放在主页的更下方的位置。可是你失去了内容平等的权利，因为智能手机版网站缺失了许多桌面版网站中的内容。

我个人不喜欢所有内容都在一个页面上的网站，这样的网站通过锚点链接访问各个内容部分，使它看起来像是单独的页面但实际并不是。用户很容易迷失在页面中，并且如果他想分享特定的内容，会发现无法简单地从浏览器地址栏复制网址，因为这个网址就是主页的网址。（圣母大学网站的主页甚至未使用真正的锚点链接，锚点链接在网址的末尾附加了额外的标识，允许你链接到页面的特定部分，比如 #admissions。）

欣慰的是对于手机用户，他们没有将所有内容放在一个页面上，因为在小屏幕上浏览那么长的页面有点困难。但我也不认可他们在一个页面上向桌面和笔记本电脑用户展示所有的

内容。这不仅容易让人晕头转向，还意味着过重的页面重量，并不是每个用笔记本电脑访问的用户都有一个快速的网络连接，特别是在校园里携带自己笔记本电脑的学生，可能并没有良好的 WiFi 信号。



图 11-7：手机上的圣母大学网站是一个完全不同的版本

11.14　总结

网站的性能应该被看作是网站设计的一部分，因为它会极大地影响到网站的用户体验。如果从项目一开始就考虑妥当，而不是试图在最后才去解决性能问题，你能把性能优化工作做得更好。

有许多工具可以用来测量性能，它们能明确指出是什么拖慢了你的网站并告诉你需要去修复什么。

你需要从编写良好的 HTML、CSS 和 JavaScript 代码开始。代码应该是简单直接不重复的。可以使用缩小工具减小文件的大小，去除多余的空白字符。

提高网站性能的一个最有效方法就是减少对服务器的 HTTP 请求数量。可以通过合并 CSS、JavaScript 或图片文件来实现此目的。第三方代码也可能产生过多的 HTTP 请求，所以一定要对添加到网站中的插件及其他代码进行仔细复核。

检查服务器设置，从避免不必要的重定向开始来设置你的网站。你还须要检查服务器是否打开了文件压缩选项并允许浏览器缓存。

JavaScript 会对网站性能造成影响，特别是那些会阻塞站点内容加载和渲染的 JavaScript 脚本。可以通过使用内联脚本、延迟加载、推迟执行来解决阻塞问题。

确保在所有的页面上只加载需要的 JavaScript 和 CSS，并在可能的情况下用 HTML/ CSS 替代脚本。

托管主机也会影响性能。确保你为自己主机所选择的公司及主机计划在性能上有良好的口碑。还可以使用内容分发网络来加速网站，并确保内容管理系统不会减慢网站的速度。

有条件地加载内容或者使用 RESS（响应式设计＋服务器端组件）都有助于提高性能。

总之，良好的性能来源于将大量有助于提升性能的方法融入到网站的创建过程中。

O'REILLY®

学习响应式设计

想要为平板、手机、笔记本、大屏幕设备，甚至可穿戴设备提供最优的用户体验？那就学习响应式设计吧。这是一本内容特别全面、讲解非常透彻的入门书。特别地，通过这本书不仅能迅速掌握响应式Web设计的基本原理，还能够从头到尾了解响应式设计的工作流程：从项目启动开始，到项目最终上线为止。

只要你的工作与创建、改造或者升级网站有关系，都应该看看这本书。换句话说，不仅是前端开发人员，设计师、产品经理、项目经理，甚至后端开发人员也可以通过本书掌握响应式设计的精髓所在。这本书基于响应式设计的前沿技术和社区经验写成，汇集了前人的智慧和最佳实践。希望读者能够“站在巨人的肩膀上”，把这本书的内容应用到未来的项目中，造福更多用户。

- 内容策略的制定应先于视觉设计
- 默认设计应针对最窄屏幕宽度进行
- 响应式Web设计的关键：HTML元素和CSS属性
- 基于设备视口宽度，通过媒体查询显示不同的CSS样式
- 处理图像、文字排版和导航
- 采用性能优化技术建立更轻量级、更快的网站

“本书内容非常全面而又不失深度，对响应式Web设计的各种概念和理论给出了极棒的阐释！”

——Tom Barker
美国康卡斯特公司
软件工程与研发部总监

Clarissa Peterson

用户体验设计师、Web开发人员，响应式网站设计咨询公司Peterson/Kandy（位于蒙特利尔）联合创始人。她经常在开发者会议及研讨会上发表关于响应式设计、移动策略和用户体验方面的演讲。

USER EXPERIENCE / DESIGN

封面设计：Randy Comer 张健

图灵社区：iTuring.cn
热线：(010)51095186转600

分类建议 计算机/程序设计/Web开发

人民邮电出版社网址：www.ptpress.com.cn

O'Reilly Media, Inc.授权人民邮电出版社出版

此简体中文版仅限于中国大陆（不包含中国香港、澳门特别行政区和中国台湾地区）销售发行
This Authorized Edition for sale only in the territory of People's Republic of China (excluding Hong Kong, Macao and Taiwan)



ISBN 978-7-115-38973-2
定价：69.00元